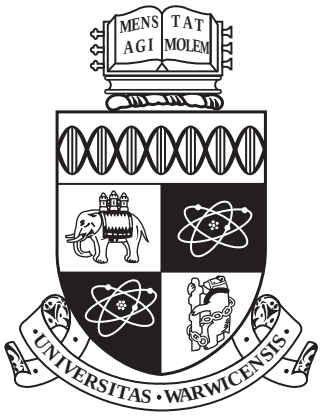


Noise-sensing Energy-harvesting Wireless Sensor Network Nodes

by

Wilson M. Tan

A thesis submitted to The University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

Doctor of Philosophy in Computer Science

Department of Computer Science

The University of Warwick

March 2016

Abstract

Noise pollution is becoming an increasing concern in many urban regions all over the world. An important step in fighting and mitigating noise pollution is its quantification. Wireless sensor networks (WSNs) can potentially help with these efforts, as they enable the simultaneous and continuous gathering of data over wide geographic regions. The need to replace batteries however makes the maintenance of such physically very large networks impractical. As an alternative to batteries, noise-sensing WSNs could also be powered by energy harvesting. While energy-harvesting WSNs have been demonstrated before, utilizing energy harvesting for powering noise-sensing WSNs still pose a significant challenge because of application's unique requirements, such as a high power consumption profile for extended periods of time. In this thesis, we address *four key areas* of research necessary on to make energy-harvesting noise-sensing WSNs possible and, more importantly, practical to use in large-scale settings. The first key area that we address is that of *new and emerging energy storage technologies*, and how current algorithms and infrastructures must be modified to take advantage of them. The second key area is that of *currently-accepted technical requirements*, and their assessment on whether they would indeed lead to the attainment of long-term goals. The third key area is that of *test methodologies* for energy-harvesting designs, and how they should be modified to facilitate validation of results between researchers. The final key area is that of *techniques and algorithms for future capabilities* that energy-harvesting noise-sending WSNs will or can have, and how we should prepare for them, even though they may not yet exist. We provide research to support all four key areas in this work and provide concrete examples for each.

Acknowledgements

An undertaking such as this would not be possible without the help and generosity of a lot of other people and institutions. As such, I would like to acknowledge...

- my supervisor, **Prof. Stephen Jarvis**, for the guidance and technical advice, as well as the connections and opportunities he made possible.
- the **Republic of the Philippines** for supporting my studies under the *Engineering Research and Development for Technology (ERDT) Program.*
- my home institution, the **University of the Philippines - Diliman**, for supporting my studies and generously allowing me to take a study leave.
- the **UK Technology Strategy Board (TSB)** for supporting our project under the *TSB Emerging Technologies Programme* (Project 131187/26835-183208, *OPV-based Energy Harvesting for Urban Noise Pollution*).
- our partners in the TSB project: Warwick's **Molecular Solar** and Cambridge's **PolySolar**, for helping make the TSB project possible, and an eventual success.
- our project partners in the **Center for Urban Science and Progress** for hosting me in New York and for providing invaluable technical advice about noise pollution in urban environments.
- **AVNET Abacus** for the gift of CBC-EVAL-09 boards.
- the 'unsung' heroes of many technical projects since the establishment of the very first research laboratories, the technicians. The Warwick DCS technical staff (**Rod Moore**, **Stuart Valentine**, and **Roger Packwood**) provided superb technical advice and help in building countless prototypes and contraptions.

Declarations

The results in Chapters 2.5.2 - 2.5.3 were previously published in the **Proceedings of the 2013 IEEE International Conference on Wireless Sensors** as the article '*Determination of maximum supportable receiver wakeup intervals in energy harvesting WSN nodes using a client-server setup* [146]'.

The results in Chapters 2.5.4 - 2.5.5 were previously published in the **Proceedings of the 2013 IFIP Wireless Days** as the article '*Self-determination of maximum supportable receiver wakeup intervals in Energy Harvesting WSN nodes* [144]'.

The results in Chapter 3.4 were previously published in the **Proceedings of the 2013 IEEE International Conference on Wireless Sensors** as the article '*Energy harvesting noise pollution sensing WSN mote: Survey of capabilities and limitations* [145]'.

The results in Chapter 2.6 and Chapter 3.5 were previously published in the **EURASIP Journal on Wireless Communications and Networking** as the article '*On the design of an energy-harvesting noise-sensing WSN mote* [143]'.

The results in Chapter 5.5 were previously published in the **Proceedings of the 2014 IEEE International Conference on Wireless Sensors** as the article '*Quality over quantity: Target identifiability in directional sensor networks* [147]'.

The results in Chapter 5.6, sans those related with 2S-DGA, were previously published in the **Proceedings of the 2015 International Conference on Computing, Networking and Communications** as the article '*A distributed heuristic solution to the target identifiability problem in directional sensor networks* [148]'.

The articles [147] and [148] were merged and extended as the article '*Heuris-*

tic Solutions to the Target Identifiability Problem in Directional Sensor Networks', and submitted for publication (December 2014) in the **Journal of Network and Computer Applications**. The article (manuscript reference number JNCA-D-15-00073) is currently being reviewed. The article covers all the results presented in Chapter 5.

A majority of the results from Chapter 4 can be found in the article '*An indoor test methodology for solar-powered wireless sensor networks*', which was submitted for publication (June 2015) to the **ACM Transactions on Embedded Computing Systems**. The article (manuscript reference number TECS-2015-0087) is currently being reviewed.

Typesetting Convention

To facilitate readability, we utilize the following typesetting convention this work

- variable names, parameters, and quantities are written using a sans serif font
- function names, as well as TinyOS commands and events are written using a `typewriter font`
- message types are written using *ITALIC LETTERS IN ALL CAPS*
- constants such as boolean vales are written using **BOLDFACE LETTERS IN ALL CAPS**

Contents

List of Figures

List of Tables

CHAPTER 1

Motivation and introduction

1.1 Motivation

Noise pollution is becoming an increasing concern in many urban regions all over the world (for example, [73]). An important step in fighting and mitigating noise pollution is its classification and quantification. Efforts being made towards this goal include cellphone applications that measure noise and noise-related legislation (such as the European Union's Environmental Noise Directive [33] and the New York City Noise Code [95]).

WSNs can potentially help with these efforts, as they enable the simultaneous and continuous gathering of data over wide geographic regions. Several WSNs for noise pollution monitoring have already been demonstrated.

The first is the work presented in [129] and [130], which features an extension board for the TelosB [123] WSN node that enables it to perform noise sensing. The extension board carries out not just noise sensing, but the output metric computation as well. The board is demonstrated to be accurate within 2 dB when compared to a professionally calibrated sound level meter (SLM). How data from a network of their nodes can be collected is presented in [35].

Another noise-sensing system, this time based on the CiNet node [42], is presented in [43]. Similar to the work presented in [129], the system is demonstrated to be accurate within 2 dB. The networking and communication component of the system is presented in [44].

While the two studies previously mentioned solely concentrate on producing measurements in dB, the work presented in [21] and [69] uses a fuzzy-logic-based method for inferring subjective noise annoyance. The system is based on a Sun

SPOT [96] WSN node.

WSN nodes (including those in the aforementioned studies) however, are usually powered by batteries which have to be frequently replaced. The length of time between battery replacements depends on the energy demand of the application: for those nodes that have energy-intensive sensors, or do lots of routing, this can be days, while for nodes that have simple sensors with very small duty cycles, this can be many months. The task of replacing these batteries on a regular basis makes the maintenance of such networks difficult.

As an alternative to batteries, WSNs can also be powered through energy harvesting. Several energy-harvesting WSNs have been demonstrated [115][131][139]; nevertheless, despite these past successes, creating an energy-harvesting WSN for noise pollution monitoring is not straightforward. The difficulty lies in the nature of the data being gathered. While a temperature-sensing WSN can probably take a reading every minute and not lose accuracy, to measure noise, sound samples have to be taken. The human ear can hear frequencies of up to 20,000 Hz - to be able to digitally reconstruct a human-audible signal without missing any frequency component, samples should then be taken at the rate of the Nyquist frequency, or around 40,000 Hz. Sampling at such a high frequency is highly energy consuming, and the processing of the data gathered is challenging to implement on resource-constrained motes.

Because the advantages of energy-harvesting WSNs are apparent and numerous (not just in urban areas, but arguably even more so in remote deployment locations), a significant amount of theoretical work has been done on what it will take to get such systems working.

[59][58][49] introduces the concept of *energy neutral operation*, a mode wherein the node only consumes what energy it harvests from the environment, thus resulting in an infinite lifetime. They provide a theoretical framework for deriving the required battery size and initial battery charge for energy neutral operation, given power supply and power consumption functions. Using the framework as a basis, a low complexity power management algorithm is also derived. The

algorithm first makes predictions about future energy supply based on past observations. Duty cycles are then assigned to slots. Depending on whether the actual harvested energy is higher or lower than the level predicted, the duty cycle assigned to the application for a given slot is then adjusted.

[9] surveys the main approaches to energy conservation in WSNs. They classify approaches into three categories: duty-cycling techniques, data-driven techniques, and mobility-based techniques. These, especially the first two, can be thought of as the 'levers' or 'knobs' available to a power management algorithm to control the power consumption of a node.

[3] surveys techniques for energy management in WSNs with energy-hungry sensors. They note that most energy management techniques focus on managing the energy consumption of the radio (communication system), as it is usually assumed that data acquisition and processing consume significantly less energy than communication. They point out that this assumption does not hold true in many real world applications. They identify two main approaches for reducing and controlling sensor energy consumption: duty cycling and adaptive sensing. Duty cycling consists of activating a sensor for only a limited period of time, and having it in an inactive state for most of the time. Such is also the strategy employed for reducing energy consumption in radios. Adaptive sensing, on the other hand, consists of changing the operation of the sensor based on the current state of the quantity being measured or entity being monitored. They note that in energy harvesting systems, the system can also take into account the energy level of the system along with the state of the environment being monitored.

Despite the progress that has already been made on the theory, significant challenges still remain in the creation of energy-harvesting noise-sensing WSNs that can practically satisfy the requirements of urban sensing efforts.

1.2 Introduction/Preliminaries

1.2.1 Wireless sensor networks: Areas of application

The past few years have seen the proliferation of research on different aspects of wireless sensor networks (WSNs): these include mote design [123], network protocols [52], and placement algorithms [135], to mention a few. With their capability of gathering geographically wide-spread data at relatively minimal (and falling) cost, WSNs have also enabled other studies. Ecologists, environmentalists and field biologists have particularly benefited from WSNs, as they enable them to observe phenomena previously unobserved due to inaccessibility of sites (rain forests, or glaciers) or mobility of targets being observed (as is the case with highly mobile animals, such as zebras) [124] [142]. Notable examples of WSNs for environmental or animal monitoring include Sensorscope [50] and ZebraNet [163]. WSNs have also found significant application in the field of agriculture. Different subfields within agriculture (and different crops) have different needs, leading to widely different technical requirements and designs. Sensors found in agricultural WSNs range from those that are found elsewhere in other applications (for example, temperature or humidity sensors), to those that are highly specific to agriculture (soil moisture sensors and phosphate sensors, for example). Studies demonstrating how WSNs can be applied in agriculture include [164] (plant nursery monitoring), [13] (vineyard monitoring) and [157] (dynamic irrigation system for cotton crops). WSNs can also potentially have applications in commercial logistics, as evidenced by CargoNet [68], which is a platform for monitoring crates and cases for supply chain management. WSNs might also someday find their way inside homes, as exemplified by the products of Nest Labs [94]. While only a few Nest products are found in most installations, a network of Nest sensors can arguably be classified as a 'wireless sensor network'.

Another area where WSNs can make significant contributions is in the increasing number of 'smart city' initiatives all over the world. While different

organizations define 'smart cities' differently [12][25][41], almost all definitions of 'smart city' agree that it will be a city that is more livable, sustainable, and resilient than the urban spaces that we have today. This will be of paramount importance as the percentage of world population that lives in urban areas continues to increase: currently at 50%, the World Health Organization estimates that this percentage might reach up to 70% by the end of the century [116]. Although smart city initiatives are almost always multi-faceted, almost all emphasize the importance of information and communication technologies (ICT) for them to achieve their goals. It is within this ICT scope that WSNs usually find their place, specifically filling the need to sense, measure and acquire data [12]. WSNs are an integral part of an 'intelligent infrastructure' [25] and a technology that 'ensures equity, fairness, and realise a better quality of city life' [12].

While the usefulness of sensor networks in smart cities may be readily apparent, it must be noted that they need not be *wireless*. Indeed, unlike in remote areas, the presence of infrastructure readily solves some of the problems faced by WSN deployments. Sensor nodes in an urban area, for instance, can be attached to lamp posts and be configured to draw power directly from the energy grid. Even the data gathered can possibly be sent to the base station through wires, using technologies such as the Ethernet. Nevertheless, there are advantages to sensors being untethered, including ease of deployment and reconfigurability. Wireless sensor nodes can be positioned in such a way that their collective effectiveness can be maximized. WSN urban deployments face unique challenges however, such as interference (since most transmit using the industrial, scientific and medical or *ISM* band, which is also widely-used by consumer electronics) and difficulty of maintenance. The latter is largely where this thesis makes a contribution.

Sensor networks have already been demonstrated of being capable of sensing parameters of interest in urban areas, including air pollution and temperature. In this thesis, we focus on a specific aspect of urban sensing: the sensing of noise

or sound levels.

1.2.2 Energy harvesting

Energy harvesting is the process of converting energy from the environment into electricity. Energy harvesting is not exclusively used for wireless sensor nodes, as they have actually been used for the generation of grid electricity for far longer, providing an alternative to electricity generated from fossil fuels. Energy harvesting or renewable energy sources have gained prominence as the world's supply of non-renewable fossil fuels (such as oil and gas) started to dwindle, and because of their deleterious effects on the environment (for instance, climate change) and harmful effects on human health (pollution).

Wireless sensor nodes can be powered through energy harvesting in several ways. Energy sources that have been successfully demonstrated include motion (vibration) [11], water flow [111], wind [18], heat [127], radio waves [158], and solar energy.

Solar energy harvesting is the most commonly utilized energy harvesting technology for WSNs, because of several reasons [60].

Firstly, solar energy harvesting is a mature technology and offers higher efficiencies, especially when compared to technologies such as radio wave energy harvesting or vibration harvesting.

Secondly, solar cells, which are used for harvesting solar energy, are cheap when compared to its counterparts in other energy sources. Solar cells also do not have moving parts, since it harvests energy through an electrochemical process. This translates to lower maintenance costs. In comparison, water flow and wind energy are harvested through an electromechanical process, and their transducers (the device which converts from one energy form to another) tend to be bulky.

Another factor that works in solar energy harvesting's favour is that its output is naturally Direct Current (DC), which is readily used by electronic circuits. In comparison, water flow and wind energy have Alternating Current

(AC) outputs, which have to be converted to DC before they can be used to power electronic circuits. Frequently the output of a solar cell must be converted to a proper voltage level before it can be used by electronic circuits. This is facilitated by DC-DC converters. However, DC-DC converters are in general simpler and cheaper than AC-DC converters.

It must be noted that for generating electricity for the electric grid, water flow energy (or hydro-electric) and wind energy dominate solar energy in terms of usage [32]. However, facilities for grid electricity generation are usually long-term investments that last decades, thus enabling the costs to be amortized over longer periods of time. In comparison, sensor nodes are generally envisioned to be cheap or even disposable devices which are easily replaced and deployed in large numbers.

Solar cell physics

Solar energy is harvested though solar cells, also known as photovoltaic (PV) cells. Solar cells function through the *photovoltaic effect*. Here we give a necessarily brief discussion of the photovoltaic effect. The physics of solar cells is discussed in greater details in books on photovoltaic engineering such as [71] and [109]. The aforementioned sources serve as our primary references for the following discussion. Readers that want to delve even deeper into the physics behind solar cell operation can consult literature on solid state devices such as [140].

Most solar cells are made from semiconductor materials. Semiconductors such as silicon have the characteristic that they are perfect insulators at absolute zero temperature, but increasingly become conductors as the temperature is increased.

Light consists of photons which carry energy (Equation 1.1):

$$\mathsf{E} = \mathsf{h} \times \mathsf{v} = \frac{\mathsf{h} \times \mathsf{c}}{\lambda} \tag{1.1}$$

where h is Planck's constant ($6.6 \times 10^{-34} Js$), c is the speed of light ($2.998 \times 10^{8} \frac{m}{s}$), v is the frequency of the photon in Hz, and λ is the wavelength of light in meters.

Exciting the electrons in a semiconductor requires a certain amount of energy called *bandgap energy*. When light strikes the semiconductor and the photons contain energy equal to or greater than the bandgap energy, the electrons are excited into a higher energy state. More specifically, *electron-hole pairs* are generated. However, this excitation of electrons cannot by themselves be harnessed to power devices, as the holes and electrons immediately recombine. To generate electricity, the movement of the electrons must somehow be guided.

Generating electricity from the electron-hole pairs require that the semiconductors be *doped*. Doping is the process of adding small amounts of impurities to the semiconductor so that it would have an excess of holes or electrons. Depending on the impurity added the semiconductor can be come *p-type* or *n-type*. Boron is an example of element added to silicon to produce p-type semiconductors, while the addition of phosphorus results in the production of n-type semiconductors.

When p-type and n-type semiconductors are placed in contact, a *junction* is formed. More importantly, since there is an excess of electrons on one side of the junction and an excess of holes on the other side, there is a consistent traffic of charge carriers across the junction. This consistent traffic leads to the generation of a built-in *electric field*.

Returning to the case of light-generated electron-hole pairs, when the electron-hole pairs are excited in the junction, the electrons are pushed by the built-in electric field towards the n-type semiconductor and the holes towards the p-type semiconductor. This results in an excess of charge carriers in each semiconductor type. The excess charge carriers then result in a voltage between the external terminals of the material (attached to each semiconductor type). If an external wire is connected between the terminals, current will flow from the p-type semiconductor towards the n-type semiconductor, generating electricity.

There are several types of junction that can be constructed, but all serve the purpose of converting the electron-hole pairs liberated by energy from photons to a flow of electrons between the device terminals.

The losses experienced (hence, efficiency) of a solar cell is affected by several factors. The type of material dictates the bandgap energy, hence the light frequency band from which it can harvest energy. The construction or geometry of the device affects the overall resistance and how efficient the device is in minimizing the number of recombinations that do not contribute the current flow between the terminals.

Types of solar cells

Solar cells are generally classified according to the materials used in their construction.

- **Crystalline silicon** [71]. Crystalline solar cells use materials that are highly crystalline in structure. For silicon-based solar cells they are the most robust and efficient. They are also the most energy-intensive in terms of manufacturing. Their manufacturing process has significant similarities with those of semiconductors used for microelectronics, but with less stringent requirements when it comes to purity and clean-room handling.

- **Thin film** [71]. Thin film solar cells use materials that do not have highly crystalline structures. They generally use less materials in their construction compared to their crystalline silicon counterparts. Their manufacturing process is also simpler and less energy-intensive, resulting in a cheaper product. They are however, also less efficient than crystalline silicon solar cells. Contrary to what their name may suggest, thin film solar cells are usually heavier than their crystalline silicon counterparts since the film has to be encapsulated between two panes of glass, compared to crystalline silicon, which only uses a single pane. Thin film solar cells are not exclusively made of silicon. Most thin film solar cells can be further classi-

fied into amorphous silicon, cadmium telluride (CdTe), or copper indium gallium selenide (CIGS) depending on the material they are made of.

- **Emerging technologies** [71][110]. It must be noted that other types of materials can exhibit the photoelectric effect discussed, not just semiconductors. Some organic materials can also generated electricity from incident light - although they work differently from the mechanism previously discussed, relying on *excitons* and not simple electron-hole pairs. Such devices are called Organic Photovoltaics, or OPVs. OPVs is very much an active research area. Driving such efforts are their potentially lower manufacturing costs compared to crystalline silicon or thin film solar cells, as well as other desirable features such as flexibility and transparency. Another factor driving their development is their environmental friendliness. The process of turning silicon into solar cells is not only energy-intensive (hence expensive), it also involves several toxic chemicals. The presence of toxic chemicals make the disposal of such devices challenging. In comparison, production of OPVs involve no such chemicals, and they can be more readily disposed. Despite the promise of emerging solar cell technologies however, most of them remain at the research stage. Compared to crystalline and thin-film solar cells, their efficiencies are lower and their lifetimes are still significantly shorter.

1.3 Energy-harvesting noise-sensing WSNs: Key areas of development

In this thesis, we propose that for energy-harvesting noise-sensing WSNs to become a reality, designers, researchers and engineers will have to focus on four key areas:

1. New and emerging energy storage technologies, and how current algorithms and infrastructures must be modified to take advantage of them;

2. A review of currently-accepted technical requirements, and an assessment on whether they will indeed lead to the attainment of long-term goals;

3. Improvements in current methodologies, specifically methodologies involved in testing designs;

4. A forward-looking view of the capabilities that a fully-functional system will or can have, with the goal of creating techniques or algorithms that will enable users to take advantage of them.

1.3.1 The key areas of development and the thesis structure

In the following, we expound on each of the key areas, and how they relate to the overall structure and contribution of the thesis.

New and emerging energy storage technologies

A system like an energy-harvesting noise-sensing WSN utilizes inputs in either knowledge or technology from several disciplines: computer science (algorithms), electrical engineering (hardware design), chemistry (energy harvesting and storage), mechanical engineering (acoustic transducers), psychology (noise perception), and medicine (health effects of noise). Relevant progress in any of the said fields have the potential of moving the field forward. Nevertheless, most of the time, new technologies pose challenges of their own: while some will work in a simple 'plug and play' fashion, directly replacing older equivalent counterparts, most do not. For these technologies, integrating them into existing systems entail the creation of new methods or algorithms, or modification of existing ones. It is therefore important that new technologies be evaluated carefully rather than simply assuming that they will have an immediate positive impact on the overall system. It will also be worthwhile to extrapolate in advance the implications of these new and upcoming technologies, so that algorithms and operating systems can prepare for them in advance. In this thesis

we identify thin film solid state batteries as a specific example of new and upcoming energy storage technologies, and discuss how they can potentially help in future energy-harvesting noise-sensing WSNs, along with the challenges or problems that must first be solved before they attain that potential.

It has been noted that the field of energy storage has advanced at a pace significantly slower than that of electronic computing devices. This has had the effect of energy storage devices taking up most of the space or volume in fully-integrated systems. Most energy-harvesting WSN nodes have relied on either supercapacitors [78] or rechargeable batteries (either Li-ion [101] or NiCd [97]) as energy storage devices. However, a new type of energy storage device is emerging: thin film solid state batteries, primarily represented by the Cymbet EnerChip [22]. Manufactured using semiconductor processing technologies, thin film solid state batteries are smaller than their coin cell counterparts [79], can handle more charge-discharge cycles than most rechargeable batteries [81], and have very low leakage compared to supercapacitors [79]. They are also more eco-friendly than supercapacitors or rechargeable batteries as they do not contain hazardous substances and are easy to dispose of at the end of their life cycle [82]. Despite these advantages however, a significant disadvantage of thin film solid state batteries is their high internal impedance [80]. The high internal impedance effectively puts a limit on the amount of current that can be drawn from the battery over short periods of time. This in turn effectively forces the node to duty cycle.

Traditionally, duty cycling has been done to conserve energy, prolong node lifetime, and to adjust the node power consumption to the amount of energy being harvested from the environment. When thin film solid state batteries are utilized, the energy storage device imposes its own duty cycling requirements. This hardware imposed-constraint (in contrast to energy-imposed constraint) is potentially detrimental to system performance. In Chapter 2, we propose ways of mitigating the said effect, both in the context of radio duty cycling (via LPL, or Low Power Listening [104]) and the more general context of operations with

high current draws (for example, sampling the microphone).

Review of currently accepted and prevalent requirements

Requirements and design goals have tremendous influence on the efforts being made towards advancing energy-harvesting noise-sensing WSNs. As such, it is worthwhile reviewing the requirements and design goals that researchers and systems designers strive to follow. It should be assessed whether widely-accepted technical requirements still faithfully correspond to the overall goal of the effort.

Most previous efforts in realizing energy-harvesting noise-sensing WSNs have aimed towards measuring the *continuous equivalent A-weighted sound level*, or $\mathsf{L}_{\mathsf{eqT}}$, because most existing noise codes specify $\mathsf{L}_{\mathsf{eqT}}$ as their primary metric. Nevertheless, our correspondence and interaction with domain experts have made it clear to us that while $\mathsf{L}_{\mathsf{eqT}}$ is useful, it is not necessarily an adequate metric, and there are certain aspects of noise as experienced by humans that it misses. Such views are also well supported in technical literature [113].

In Chapter 3, we demonstrate that $\mathsf{L}_{\mathsf{eqT}}$-measuring systems are challenging to implement as energy-harvesting devices. We also propose an alternative design that does not measure $\mathsf{L}_{\mathsf{eqT}}$, but can still be useful in urban settings. The alternative design is better-suited to being powered by energy harvesting than designs geared towards measuring $\mathsf{L}_{\mathsf{eqT}}$.

Test methodologies

The methodologies used in designing and testing energy harvesting-powered systems must also be advanced if progress is to be made towards realizing energy-harvesting noise-sensing WSNs. We note that there is no universal standard for testing energy-harvesting WSN nodes in terms of software/algorithms (duty cycle, etc.) and hardware parameters (choice of microcontroller, size of the energy storage device, etc.). This makes verification and design comparison difficult, if not impossible. As such, hardware and software parameters for most designs are simply estimated, and the actual deployment serves as the test

itself. The main problem with actual deployments is their non-repeatability. By 'repeatability' we mean the ability to test different algorithms, parameters, and hardware designs *several* times under *similar* energy-relevant conditions, leading to fair and verifiable comparisons. Because actual deployments are non-repeatable, if mistakes are made during the estimation of parameters, then it will already be too late or too costly to change them. The test methodology that comes closest to being a true predeployment methodology is the test utilized in [63], where a TI eZ430-RF2500-SEH [153], with a newly designed power management algorithm, is exposed to a lamp that is then manually turned on and off. This is still not a satisfactory approach however, since solar panels can be easily put in states they will never be in actual outdoor conditions.

In Chapter 4, a methodology for indoor testing actually-built solar-powered WSN nodes (and networks of such nodes) is presented. The methodology is based on insights from the field of photovoltaic cell design and astronomy. The methodology enables repeatable tests and frees the tests that could be performed from geographic and seasonal constraints. The test methodology is carried out using a test apparatus which 'simulates' the effect of an astronomical theory-predicted sunlight pattern (at a particular place and time) on a solar panel. A *generic* design for such an apparatus is presented in Chapter 4, along with a *specific* design which is used in the construction of an actual apparatus. A series of experiments - the likes of which are important prior to an actual WSN deployment - are carried out to demonstrate the potential of the test methodology and implemented test apparatus.

Algorithms for future capabilities

Future fully-operational energy-harvesting noise-sensing WSNs can or will have capabilities that we are simply not designing or taking account of at present. None of the existing work for instance, has taken into account localization or identification of noise sources. Absence of interest and research in the area may be justified, since most previous work has dealt with single nodes and not fully-

deployed networks of such nodes. Nevertheless, a deployed energy-harvesting noise-sensing WSN can potentially perform noise source localization, provided that the nodes are fitted with directional microphones. The network then effectively becomes a *directional sensor network*, and the problem, one of target identification or localization. In Chapter 5, we present heuristic algorithms (both centralized and distributed) for orienting directional sensor networks in such a way that both the number of targets covered and their identifiabilities are taken into account, and maximized. These algorithms can be readily applied to noise-sensing WSNs (provided that they are fitted with directional microphones), enabling them to localize or identify noise sources.

1.3.2 The remainder of the thesis

Specific contributions to each of the four areas are discussed in Chapters 2-5. Because of the diversity of the areas and for increased readability, related and possible future work are also discussed in Chapters 2-5, after the specific contributions. Chapter 6 concludes the thesis.

CHAPTER 2

New and emerging energy storage technologies: Mitigating hardware-imposed duty cycling constraints

2.1 Overview

New and emerging energy storage technologies can potentially help in improving the performance of energy-harvesting WSNs (not just those designed for noise-sensing). Some of these technologies however cannot be readily adopted without first modifying existing algorithms or introducing new techniques, as doing so may actually end up hindering, instead of helping, performance.

In this chapter, we provide a specific example of such a technology with thin film solid state batteries, primarily represented by the Cymbet EnerChip [22]. As mentioned in Chapter 1, thin film solid state batteries offer the advantages of smaller form factor, lower leakage, and more charge-discharge cycles, compared to most rechargeable batteries. However, thin film solid state batteries also impose additional constraints on the duty cycle that can be adopted by the WSN node. This additional constraint is potentially detrimental to performance. In this chapter, we present techniques that minimize or counteract such an effect.

2.2 Introduction and motivation

One of the biggest issues in wireless sensor networks is energy. The issue is even more critical in energy-harvesting wireless sensor networks, which usually have access to energy supply that is intermittent or orders of magnitude lower than that of batteries. In non-energy harvesting WSNs, energy conservation translates to longer lifetimes (or intervals between battery replacements) while

in energy-harvesting WSNs, energy conservation and management are at times crucial for the system to operate at all.

A common strategy for conserving energy in WSNs is duty cycling, wherein the radio (and sometimes, along with other components) is regularly turned on only for a small period of time, after which it goes back to an inactive state. The duty cycle of a system has generally been considered to be a function of the energy supply. In systems powered by energy harvesting, the duty cycle depends on the amount of energy that can be harvested from the environment, desired lifetime for the system, and the energy consumption. Rarely discussed in the literature however, are the limitations imposed on the duty cycle by the *energy storage device*. As will be discussed, certain types of energy storage devices place limitations on the duty cycle achievable, independent of the three aforementioned factors. Such limitations can have performance-limiting consequences.

In this chapter we discuss the implications of such storage device-imposed limitations both when what is being duty cycled is the radio and other components with similarly high energy consumption. We also discuss ways of mitigating the effects of the said limitations on performance. A recurring theme in this chapter is how the limits can be frequently derived or calculated using computational-analytical methods, but the limits produced by these methods are frequently conservative. The limits derived by the computational-analytical methods can then be *ignored* up to a certain extent, leading to better performance for the system. The *real* and absolute limits can be derived via empirical methods, but experimentation is tedious and labour-some. Nevertheless, the system itself can be designed to empirically derive the limits automatedly, thus leading to better performance, and not requiring human intervention.

We note that in this work in general we are trying to *increase* the duty cycle that an energy harvesting-powered WSN node can have, since higher duty cycles translate to more or longer (more informative) sensor readings.

Chapter 2.3 discusses the components and physical setup that will be used

in this chapter (and some chapters that will follow). Chapter 2.4 elaborates on how certain types of energy storage devices place limitations on the duty cycle. Chapter 2.5 discusses how the radio (and communication) is affected by the said limitations, and what can be done to maximize the performance of systems affected by the said limitations. The case for generic components (not just the radio) that have significant energy consumption is covered in Chapter 2.6.

2.3 Physical setup

For our work, we use the ultra-low-power wireless sensor module ('mote') TelosB [123]. TelosB was designed at UC Berkeley with three goals in mind: minimal power consumption, ease of use, and increased software and hardware robustness. It uses a 16-bit Texas Instruments MSP430 microcontroller [51][151] and a 2.4 GHz IEEE 802.15.4 compliant RF transceiver Chipcon CC2420 radio [155]. The TelosB motes that we use for this work were manufactured by Advanticsys [1] and marketed as CM5000 motes. Our TelosB motes run TinyOS, or Tiny Microthreading Operating System [47][103].

For the energy harvesting component of our setup, we utilize the CBC-EVAL-09 [23]. The CBC-EVAL-09 is an evaluation kit manufactured by Cymbet Corporation (Elk River, MN, USA). It features several energy-harvesting transducers, along with the EnerChip EP CBC915 Energy Processor [83] and the EnerChip CBC51100 100 μAh solid state battery module (with two EnerChip CC CBC3105 [22] solid state batteries connected in parallel).

The EnerChip EP CBC915 Energy Processor [83] serves as an interface between the transducers and the energy storage device. It employs advanced maximum power point tracking algorithms, constantly matching the output impedance of the energy-harvesting transducers, thus ensuring high-efficiency energy conversion. The EnerChip EP CBC915 Energy Processor also facilitates communication with the microcontroller, providing information such as state-of-charge estimates and a calibration function.

2.4 The energy storage device and the duty cycle

The amount of current that can be drawn from an energy storage device is limited by the device's internal impedance. To compensate for this, a capacitor is usually inserted between the energy storage device and the load. This capacitor is also called the *boost capacitor*. In this section, 'boost capacitors' will also sometimes be referred to as simply 'capacitors'. It must be noted that since the electrical charge has to be transferred from the energy storage device to the capacitor, it is still impossible to supply a high level of current to the load for indefinite periods of time.

For the same level of current draw, a longer draw period will necessitate using a larger capacitor to store a greater amount of electrical charge from the primary energy storage device. However, a larger capacitor also takes longer to charge - therefore, the intervals *between* current draws, which we also call *charge time*, will also increase.

The relationship between the capacitor size, the level of current draw (in milliamperes), the length of the current draw, and the interval between draws can be derived analytically and computed: for instance, the energy storage system that we utilized for our work, the EnerChip CC CBC3105 [22], specifies through an application note [80] a formula for determining the capacitor size needed for supporting a specified level of current draw for a specified length of time. We state this in Equation 2.1. The equation for $\mathsf{R_{Load}}$, which is a variable in Equation 2.1, is defined in Equation 2.2. Equation 2.1 and Equation 2.2's variables are defined in Table 2.1. For variables whose values remain constant across different computations, their values are specified in Table 2.2.

$$\mathsf{C} = \frac{\mathsf{t_{Draw}}}{\mathsf{R_{Load}} \times ln(\frac{\mathsf{V_{Max}}}{\mathsf{V_{Min}}})} \tag{2.1}$$

$$\mathsf{R_{Load}} = \frac{\mathsf{V_{out(average)}}}{\mathsf{I_{pulse}}} \tag{2.2}$$

Table 2.1: Equation 2.1 - Equation 2.4 variables

Variable	Description
C	capacitance of external capacitor in parallel with the battery
t_{Draw}	length of time that the current will be drawn
R_{Load}	load resistance
V_{Max}	final capacitor voltage that must be attained before next current draw
V_{Min}	initial voltage when charging begins
t_{Charge}	capacitor charge time
R_{Bat}	battery resistance
V_{Chg}	applied charging voltage on the capacitor
$V_{out(average)}$	average voltage across the load during the current draw
I_{pulse}	level of current draw (in *Amperes*)

Table 2.2: Equation 2.1 to Equation 2.4 variables and values

Variable	Value
V_{Max}	3.8 V
V_{Min}	1.8 V
R_{Bat}	7000 ohms
V_{Chg}	4.1 V
$V_{out(average)}$	3.4 V

The formula for the charge time of a given specific capacitor is also given by [80], and is stated here in Equation 2.3. Equation 2.3's variables are also defined in Table 2.1.

$$\mathsf{t_{Charge}} = -\mathsf{R_{Bat}} \times \mathsf{C} \times ln\left(\frac{\mathsf{V_{Max}} - \mathsf{V_{Chg}}}{\mathsf{V_{Min}} - \mathsf{V_{Chg}}}\right) \tag{2.3}$$

Solving Equation 2.1 and Equation 2.3, for example, a system utilizing a 5,000 μF capacitor should be able to support a current draw of 20 mA for 210 ms with a charge time of 45 s.

Defining duty cycle as the proportion of time the system is awake or active in a cycle, we now have Equation 2.4

$$dutycycle = \frac{\mathsf{t_{Draw}}}{\mathsf{t_{Draw}} + \mathsf{t_{Charge}}} \tag{2.4}$$

Substituting Equation 2.1 and Equation 2.3:

$$dutycycle = \frac{\mathsf{R_{Load}} \times ln\frac{\mathsf{V_{Max}}}{\mathsf{V_{Min}}}}{\mathsf{R_{Load}} \times ln\frac{\mathsf{V_{Max}}}{\mathsf{V_{Min}}} - \mathsf{R_{Bat}} \times ln\frac{\mathsf{V_{Max}} - \mathsf{V_{Chg}}}{\mathsf{V_{Min}} - \mathsf{V_{Chg}}}} \tag{2.5}$$

It must be noted that the only variable defined by the load which appears in Equation 2.5 is $\mathsf{R_{Load}}$. Referring to Equation 2.2, only $\mathsf{I_{pulse}}$ determines the duty cycle of the system. The capacitor, in particular, does not figure in Equation 2.4. A larger capacitor will enable a longer draw time, but its ratio to the sum of the charge time and the draw time (the *total* period) will always be the same.

In some applications, the length of the current draw may be user-definable. An example of such a load will be the microphone. To have a longer draw time, we may opt for a larger capacitor, but the interval between draws will proportionately increase as well. In other applications, the length of the current draw is already defined. An example of such a load is a radio which has to send a packet to a receiver which is duty cycling using low-power listening (*LPL* [104]). If the wake-up interval of the receiver has already been decided or set, the designer has no choice on the size of the capacitor that must be installed on

the sender.

Traditionally, duty cycling is seen as a necessity imposed by the limited amount of harvested energy. As can be seen in Equation 2.5 however, the hardware (through the energy storage device) imposes its own cap on the duty cycle feasible. It is interesting to note that neither the capacity of the energy storage device nor the solar panel output figures in Equation 2.5. Therefore, using larger batteries or larger solar panels will not help in increasing the limit imposed by Equation 2.5.

It must be noted that the level of current that can be drawn from *any* energy storage device is limited both in magnitude and duration. However the internal impedance of 'traditional' energy storage device as NiCd rechargeable batteries or supercapacitors are usually so low that they can support high levels of current draw (higher than that which most WSN nodes usually require) for durations effectively limited only by the amount of charge stored in them. As such, they do not impose any constraints on the duty cycling of the systems they power.

2.5 Mitigating performance degradation due to hardware-imposed duty cycling constraints in radio communications

TinyOS, an operating system for WSN motes, comes with the low-power listening (*LPL* [104]) protocol as its default duty cycling mechanism for the radio. As the name implies the mechanism is mainly listener-centric: provided that a receiver can tolerate certain delays between packet generation and packet reception, LPL can result in orders of magnitude less power consumption - on the receiver side.

LPL works by having the receiving node wake up every time a wake-up interval elapses and sample the channel for a short period of time. Upon detecting activity in the channel, the node stays awake for a bit longer; if not then the

node goes back to sleep, only to awaken again after another wake-up interval has elapsed. If the activity in the channel is due to another node sending to the duty cycling node, the duty cycling node will further stay awake to receive the packet.

LPL is designed to asynchronously operate; hence, a sending node does not know when a duty cycling receiving node will wake up. Thus, to ensure that it will find the receiving node awake, upon starting a send operation, it will try to repeatedly attempt to send the packet until the packet is acknowledged by the receiver or a time interval equivalent to the receiving node's wake up interval elapses, whichever comes first. A send attempt consists of a packet transmission followed by a period in which the sender waits for an acknowledgement from the receiver. Each attempt is separated from each other by a relatively short back-off period. Acknowledgements are sent by the receiver upon successfully receiving the packet. Upon receiving an acknowledgment, the sender will stop attempting to send the packet in a *unicast* operation. In a *broadcast* operation, since there are several receivers which could be waking up at different periods in time, the sender will keep attempting the sends until a time period equivalent to the longest receiver wake-up interval among the set of wake-up intervals of the intended receivers has elapsed (it is assumed that this value has been properly set by the application developer in the sender beforehand).

LPL operation is shown in Figure 2.1. In Figure 2.1, the first attempt of the sender failed (since the receiver was still sleeping when it occured) while the second attempt succeeded.

From Figure 2.1, it is apparent that while a greater wake-up interval results in greater energy savings for the receiver, it results in potentially greater energy consumption per send operation for the sender. In the diagram, a longer wake-up interval can *potentially* result in a longer A-labelled section, ultimately depending on when the send operation started and when point B will occur.

If we assume that the sender mote is utilizing an energy storage device with high impedance (such as that discussed in Chapter 2.4), for the same level of

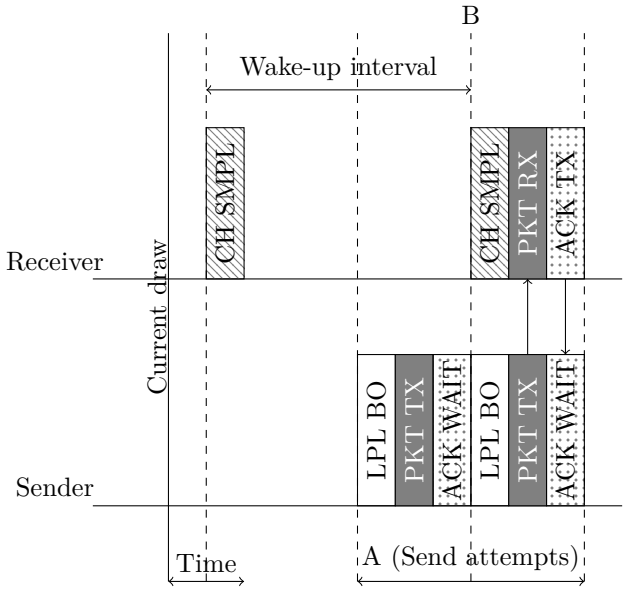


Figure 2.1: LPL operation. **CH SMPL** - channel sampling; **LPL BO** - LPL back-off; **PKT TX** - packet transmission; **ACK WAIT** - waiting for acknowledgement; **PKT RX** - packet reception; **ACK TX** - acknowledgement transmission. Durations not drawn to scale.

current draw, a longer draw period will necessitate using a larger capacitor to store a greater amount of electrical charge from the primary energy storage device (Equation 2.1). However, a larger capacitor also takes longer to charge; therefore the intervals *between* current draws will also increase (Equation 2.3).

In a sender-receiver pair that is using LPL and an energy storage device with high impedance, there is a dependency between the wake-up interval of the receiver and the capacity of the boost capacitor interfacing the sender and its energy storage device. The sender will *not* be able to support any arbitrary value chosen by the receiver as a wake-up interval - the *maximum* receiver wake-up interval which can be supported by the sender depends on the size of the sender's boost capacitor. The boost capacitor should be able to supply current that can drive the radio for the duration of the receiver wake-up interval (as the sender radio will have to be kept on for this long during a worst case scenario; referring to Figure 2.1, this corresponds to when the sender starts sending just as the receiver has gone to sleep). By the same token, the size of the boost

capacitor also affects the sender mote's sending rate, or the send interval, since the boost capacitor has to be charged. In summary, a larger boost capacitor enables a sender to support longer wake-up intervals on the receiver side, but decreases the sender's sending rate.

In order to optimize the operation of a WSN (or even just a sender-receiver pair), it is important for designers to understand and take into account the relationship between these three values (the capacitor size, the receiver wake-up interval, and the sender send interval). To derive the three values, one can utilize Equations 2.1-2.3.

To test the accuracy of the values derived using these equations, we conduct two experiments. In the first experiment, we set up an actual send-receive pair with the sender being powered by an energy harvesting system interfaced to the mote with a 5000 μF capacitor. The send interval is set to 45 seconds, and the wake-up interval of the receiver is set to different values ranging from 100 milliseconds to 1000 milliseconds. We allow the setup to run for a time period that permits 20 packets to be transmitted. The number of packets successfully transmitted versus the receiver time intervals are shown in Figure 2.2. As can be seen in the figure, as expected, after a certain point the sender mote is no longer able to transmit any packets to the receiver. Datalogger and oscilloscope readings indicate that at sufficiently long enough receiver wake-up intervals, the sender mote actually restarts. This is because all the charge in the capacitor is drawn by the node and the voltage across the capacitor drops below the point required to power the mote (around 2.7 V). However, the maximum receiver wake-up interval that the sender mote is able to support is shown to be beyond the computed value of 210 milliseconds, indicating that the value derived through computation is overly conservative. That the equations give conservative values is not surprising; Equations 2.1-2.3 are based on equations from electronic and circuit devices theory. While the mathematical models for electrical and electronic components can be quite accurate, they do not always perfectly predict circuit behaviour. Some of the parameters provided by the

datasheet and used as inputs to the equations are probably highly conservative estimates as well, ensuring that the outputs of the equations will *always* work.

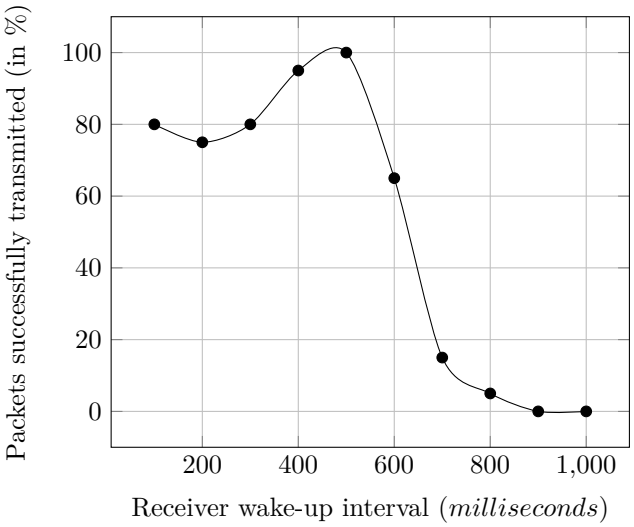


Figure 2.2: Actual transmission test, receiver wake up interval varied. Send interval is 45 seconds.

For the second experiment, we utilize the same physical setup, but instead of varying the receiver wake-up interval, the send interval is varied from 0 to 45 seconds, and the receiver wake-up interval is constantly set to 200 milliseconds. We allow the setup to run for an interval of time that is sufficient for 26 packets to be sent. Part of the results are plotted in Figure 2.3. As expected, at very short send intervals, the sender is not able to send any packets at all. This can be attributed to the node trying to draw charge from an insufficiently charged capacitor, inducing the capacitor voltage to drop, and inducing a restart. However, the sender is able to successfully send packets consecutively at a send interval that is significantly lower than the theoretical minimum of 45 seconds. Again, the value given by the computation is overly conservative.

It is important that accurate limits for the receiver wake-up interval and send interval be used if the utility of the system is to be maximized. A longer receiver wake-up interval translates to more energy savings for the receiver, so it is important that the accurate supportable limit of the sender be known. A

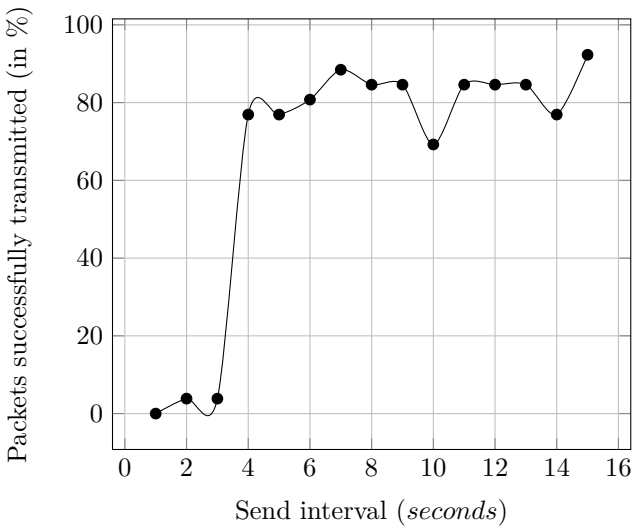


Figure 2.3: Actual transmission test, send interval varied. Receiver wake-up interval is 200 ms. All results beyond 15 seconds are equal or very close to 100%.

shorter send interval translates to more readings that can be done by the sender, and finer temporal granularity for the data gathered.

All these motivate the work done in this subchapter. Our experiments show that the computational method gives overly conservative values for the receiver wake-up interval and send interval. Experiments (such as the ones that produced Figure 2.2 and Figure 2.3) are accurate, but are obviously cumbersome and labour-intensive to perform. As an alternative to manual experimentation, we create two protocols which will enable a mote to determine through autonomous experimentation the maximum receiver wake-up interval that it can support. The first protocol is based on a client-server architecture, while the second protocol enables the node to determine the limit by itself. We note that both protocols try to find the assumed receiver wake-up interval for which the sender will shutdown or restart - we take the sender shutting down or restarting as the ultimate indicator of the assumed receiver wake-up interval no longer being supportable.

In the protocol based on the client-server architecture, one node (the *server*)

observes another (the *client*) as it tries out different assumed wake-up intervals. The technique has the advantage of being able to define supportability in terms of the Packet Reception Ratio (PRR). It must be noted that the PRR is dependent on how many packets are used to measure it. The more packets that are used in measuring the PRR, the more accurate it is, but the more energy- and time-consuming the measurement process is. To discern the most cost-effective number of packets, the protocol is tested with different *measurement window sizes*. We define the 'measurement window size' or simply 'window size' as the number of packets sent as a set when testing a wake-up interval value.

An obvious weakness of the technique is the need for another node: this may not be feasible in setups where the node has no neighbours, as is the case in networks where data is pulled or muled out of isolated sensor nodes. Even if the node wanting calibration has a neighbour, the technique requires that the neighbour must not be duty cycling for it to be able to act as a server. Thus, this technique is of limited use in a *purely energy harvesting* wireless sensor network.

The second protocol relies on non-volatile memory to enable a node to autonomously (without the assistance of another node) determine the maximum wake-up interval that it can support. Non-volatile memory is used in calibration by having the mote store the wake-up interval that was just successfully tested. If the node fails to shutdown after a calibration loop (the sequence of repeated sends at a single wake-up interval value), the wake-up interval that was just tested is stored in the non-volatile memory, and a test commences for a longer wake-up interval. Once the node restarts, it will then check the non-volatile memory, which will now contain the longest wake-up interval value that was successfully tested.

In both protocols, we would refer to the process of determining the maximum receiver wake-up interval that a node can support as a *calibration process*.

Table 2.3: Commands and events, client side

Command/Event	Command/Event name	Description
command	`calibratewakeupinterval`	called to start calibration process
command	`getpendingcalibration`	called to get results from server
event	`novolunteers`	signalled when no server volunteered
event	`calibrationfinished`	signalled when calibration finished successfully
event	`result_received`	signalled to indicate result of calibration

2.5.1 Methodology

Throughout the experiments, the EVAL-09 board is continuously exposed to light of 1300 lx+ in brightness. The light is supplied by a General Electric 240 V 60 W bulb, with the brightness controlled with a Transcension SDC-6 6-channel DMX controller connected to a Soundlab Dimmer Pack. Before running any experiment, the system is also exposed to the light source for 50 minutes, the approximate time it takes to charge up the EnerChip, according to the datasheet [23].

2.5.2 Client-server architecture-based protocol: Description

The client-server architecture-based protocol is implemented through two TinyOS components to which user level applications can wire to. The two components are *EHCalibrate* for the client-side functionality, and *EHCalibrateSERVER*, for the server-side functionality. The commands and events associated with EHCalibrate are tabulated in Table 2.3, while those for EHCalibrateSERVER are tabulated in Table 2.4.

EHCalibrate has two commands associated with it: `calibratewakeupinterval`,

Table 2.4: Commands and events, server side

Command/Event	Command/Event name	Description
command	`startserving`	called when the mote wants to start acting as a server
command	`stopserving`	called when the mote wants to stop acting as a server

Table 2.5: Input parameters, `calibratewakeupinterval`

Parameter	Description
sendinterval	send interval to use
lowerbound	shortest receiver wake-up interval value to test
upperbound	longest receiver wake-up interval to test
resolution	increment steps when changing interval under test

which starts the calibration process, and `getpendingcalibration` which fetches the result of the calibration process from a server. `calibratewakeupinterval` has four input parameters, tabulated in Table 2.5.

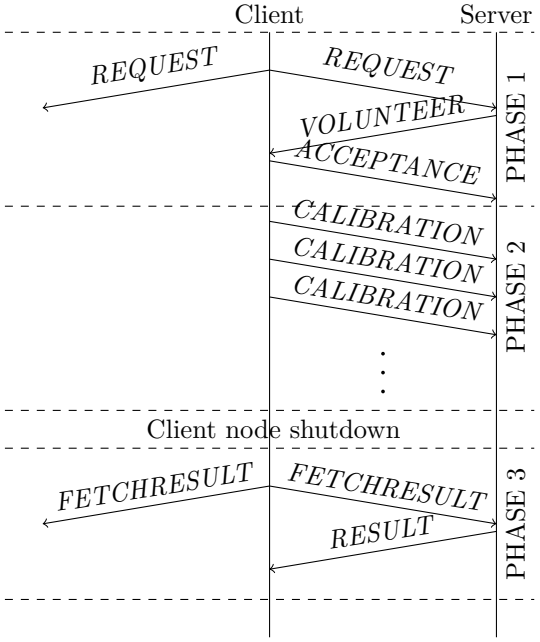


Figure 2.4: Protocol operation diagram.

Typical protocol operation is shown in Figure 2.4. A typical operation can be thought of as having three phases:

1. **PHASE 1.** Protocol operation begins by having the client-side application-level code call EHCalibrate's `calibratewakeupinterval` command. The client then broadcasts a packet requesting for server volunteers. Available servers that will receive the request will then respond with a *VOLUNTEER* packet. The client will choose among the servers that volunteered, and then send a destination-specific *ACCEPTANCE* packet. If no volunteer responds to the *REQUEST* packet, EHCalibrate will signal the `novolunteers` event.

2. **PHASE 2.** Shortly after sending the *ACCEPTANCE* packet, the client starts sending out the actual calibration packets to the chosen server. After sending a specific number of packets, the assumed receiver wake-up interval is incremented by the value specified through the input variable resolution (see Table 2.5), and a new set of packets is sent again. The process is repeated until the client node shuts down or until the upper bound value specified in the `calibratewakeupinterval` command is reached. If the latter happens, EHCalibrate signals the `calibrationfinished` event.

   While the client is sending the calibration packets, the chosen server just receives the packets and takes note of the assumed receiver wake-up interval at which the packets are being sent (the value is also contained in the calibration packet). In normal LPL operation, upon receiving the packet, the receiver (the server in our case) sends an acknowledgement so that the sender (the client in our case) will cease sending. For our protocol, we configure the motes to operate without acknowledgements. This ensures that the client will always attempt to send a packet for a period of time that is equivalent to what it assumes the server's wake-up interval is. Ideally, the motes will operate without acknowledgement while calibrating, but utilize acknowledgements for all other send operations. Another way

this can be implemented is by having the client node *broadcast* the calibration packets. If the client broadcasts the calibration packets however, the other servers that were not chosen must be configured to ignore the calibration packets.

Our earlier experiments showed that PRR degradation due to overly long receiver wake-up interval values is not abrupt: this can also be seen in Figure 2.2. To make sure that a certain receiver wake-up interval value is indeed perfectly supportable by the client, the server requires that *all* packets in a set be successfully transmitted before it considers the receiver wake-up interval as 'supportable'.

3. **PHASE 3.** The third phase is initiated by having the client application code call EHCalibrate's `getpendingcalibration` command. It is recommended that the application code call this upon boot-up, so that the results of the calibration can be collected after a client node shutdown. After a `getpendingcalibration` command is called, the client broadcasts a *FETCHRESULT* packet. A server that receives the *FETCHRESULT* packet then checks its buffer to determine whether it has recently calibrated the client. If it has, it will then send a *RESULT* packet containing the highest supportable wake-up interval value that it observed. Reception of the *RESULT* packet at the client side will then trigger the signalling of the `result_received` event, with a validity flag of **TRUE**. If after a certain period of time the client receives no *RESULT* packet, `result_received` will still be signalled, but the validity flag will have a value of **FALSE**.

   It must be noted that it is *not* necessary for the client node to shutdown during the calibration process for the protocol to work. Indeed, there are cases wherein the client does not shutdown but some of the values (in particular the higher ones) in the range that it tested were no longer supportable. Due to the server's strict requirement that all packets must

be successfully transmitted for a receiver wake-up interval value to be considered 'supportable' (see PHASE 2), the protocol will still be able to detect this. We therefore recommend that the user-level application call `getpendingcalibration` after a `calibrationfinished` is signalled. This will certainly result in a `result_received` being signalled later on. The client application code can then compare the result from the server with the upper bound of the range it just tested. If they are equal, then the entire range is supportable and the application code may opt to test a higher range of values. If they are not equal then the maximum supportable receiver wake-up interval is within the range and was detected by the server.

We note that the client-server architecture-based protocol will only work in a setup with *bidirectional* links. It does not take into account unidirectional or intermittent links which become an issue when there is a significant distance between the sender and the receiver. The presence of of unidirectional or intermittent links can cause the server to have an erroneous count of packets that the client is able to send without shutting down - the unreceived packets are now possibly due to the channel being lossy or intermittent rather than the assumed receiver (server) wake-up interval no longer being supportable by the sender (client). To minimize the effect of unidirectional or intermittent links, clients can be programmed to be able to choose servers that are closest to them, or servers from which they can receive the strongest signals. WSN nodes can measure signal strength through the Received Signal Strength Indicator (RSSI) value that many radios make available with each packet received. Alternatively, given that client would do most of the sending and the server most of the receiving, the RSSI reading on the *server* side can be taken into account instead of the client side. In this scheme, each server will report (through the *VOLUNTEER* packet) the RSSI reading it got upon receiving the *REQUEST* packet. The client will then choose the server with the highest reported RSSI value. We emphasize however, that this scheme still does not protect the system against links

going down temporarily, hence, the requirement that the links be bidirectional.

2.5.3 Client-server architecture-based protocol: Results

The client-server architecture-based protocol is tested using two nodes designated as client and server, wired to EHCalibrate and EHCalibrateSERVER, respectively. The client application code is designed to immediately call the `getpendingcalibration` command upon booting. This will result in a `result_received` event being signalled. If the result from the event is not valid (as is the case if it is the first time that the client is booting up), the application code then calls the `calibratewakeupinterval` command.

Two things can happen after this point: either the calibration will cause the node to shutdown, or it does not shutdown and the `calibrationfinished` event is signalled.

In the former, the node will boot up again, but unlike before, `result_received` will now have a valid value. The node will then proceed to use the result in sending packets to a base station. The packets that it sends to the base station also contain the calibration result value. This way, we are able to monitor what the actual calibration result is, and whether it is effective (as will be evidenced by the client being able to repeatedly send packets to the base station for a continuous period of time).

If the node does not shutdown, the `getpendingcalibration` command is called. This should result in a `result_received` event with a valid result value. The node will compare the calibration result with the upper bound value it used earlier when it called `calibratewakeupinterval`. If they are equal, the node calls `calibratewakeupinterval` again, but with the lower bound and upper bound values now incremented. If the two values are not equal, the node will then proceed to use the result value in sending to the base station.

We test three capacitor values: 3000 μF, 4000 μF, and 5000 μF, and utilize 30 seconds as the send interval. We use a resolution of 50 ms, and the lower bound and upper bound values are defined as 0 ms and 1000 ms, respectively.

An important parameter of the protocol is the measurement window size. We test four measurement window sizes: 2, 4, 6, and 8.

The results given by the calibration protocol, and the values given by the computational method, are plotted in Figure 2.5. From Figure 2.5, it is apparent that in all capacitor values tested, the actual receiver wake-up interval that can be supported by the sender (client) is much greater than the value given by Equation 2.3. Figure 2.5 also shows that the measurement window sizes tested gave very consistent limit values - the results given by the different setups varied by no more than a single resolution step for all capacitor values tested. Thus, using a measurement window size of 8 holds no accuracy advantage over a measurement window size of 2 or 4.

It is important that the measurement window size be minimized as much as possible, as larger window sizes translate to longer calibration processes. This means greater energy consumption and longer disruption of normal network operation. The times taken by the experiment setups to finish are plotted in Figure 2.6. Note that the times plotted in Figure 2.6 are greater than the actual calibration times; they are actually the time elapsed between the connection of the client node to the energy harvesting system and the reception of the base station of the first post-calibration packet from the client node. Nevertheless, the application-level code component is fairly constant across different setups, thus the differences between the times plotted in Figure 2.6 are caused by the calibration process (protocol-level component).

2.5.4 Mote-autonomous protocol: Description

The mote-autonomous protocol works by having the mote send out packets with a specific assumed receiver wake-up interval. Every time it successfully finishes the operation, it records the value of the just-tested wake-up interval in the non-volatile random access memory (NVRAM), which is a flash memory for the TelosB WSN node. After recording the value, a longer wake-up interval is then tested. The process repeats itself until the node encounters a wake-up interval

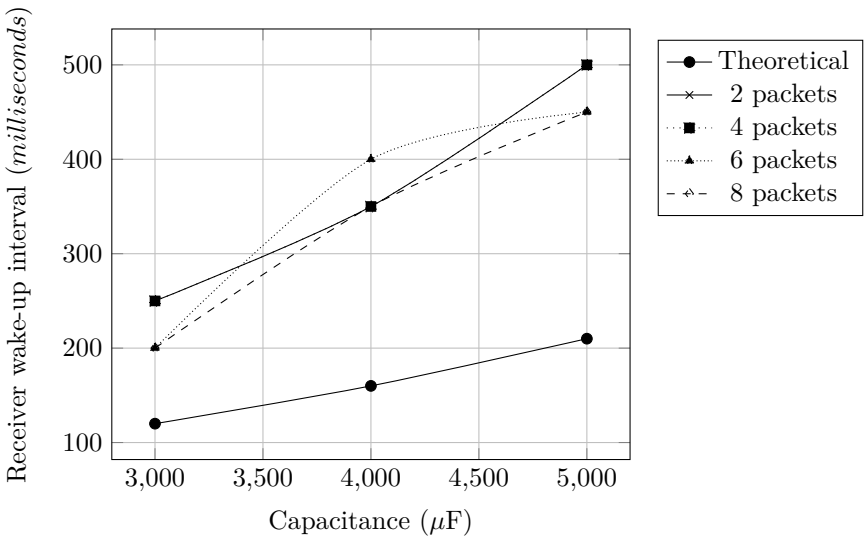


Figure 2.5: Maximum supportable wake-up interval values: theoretical vs experimental (measured by calibration protocol).

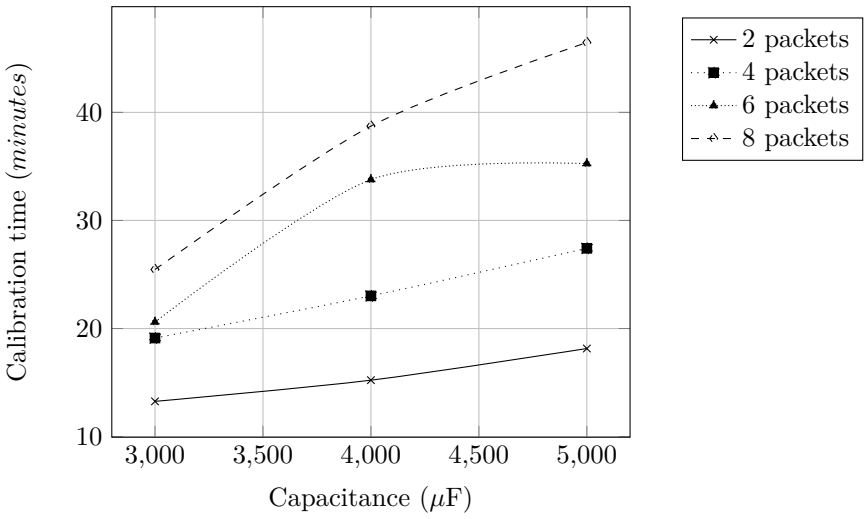


Figure 2.6: Experimental times for different setups.

it can no longer support, and will thus induce it to shutdown and restart. After booting up from a reset, the node can then read the NVRAM, which should now contain the highest wake-up interval it was able to support. The protocol is implemented through a TinyOS component to which user-level applications can connect to, *EHCalibrateSELFC*. The commands and events associated with

Table 2.6: Commands and events, client side

Command/Event	Command/Event name	Description
command	`calibratewakeupinterval`	called to start calibration process
event	`calibrationfinished`	signalled when calibration finished successfully

Table 2.7: Input parameters, `calibratewakeupinterval`

Parameter	Description
sendinterval	send interval to use
lowerbound	lowest receiver wake-up interval value to test
upperbound	highest receiver wake-up interval to test
resolution	increment steps when changing interval under test

Table 2.8: NVRAM variables, `calibratewakeupinterval`

Variable	Description
valid (boolean)	indicates whether other NVRAM variables contain valid values
wakeupintervalvalue (integer)	last wake-up interval value successfully tested
sendintervalvalue (integer)	last send interval value successfully tested

EHCalibrateSELFC are tabulated in Table 2.6.

EHCalibrate has a single command associated with it: `calibratewakeupinterval`, which starts the calibration process. `calibratewakeupinterval` has four input parameters, tabulated in Table 2.7. Table 2.8 tabulates the variables stored in the NVRAM.

Typical protocol operation is presented in Algorithm 1. Typical operation can be thought of as having three distinct phases:

1. **PHASE 1.** Protocol operation begins by having the application-level code call EHCalibrateSELFC's `calibratewakeupinterval` command. The protocol-level code then proceeds to read the NVRAM (Line 2), specifically checking for the value of the variable valid (Line 3). The variable valid

indicates the validity of the variables contained in the NVRAM. The variables are considered valid if they are produced as the result of a calibration process (hence, post-programming the initial value of valid is **FALSE**), and if the variables were not read before (in a sense, the variables, which contain the results of the calibration process, can only be 'used' once). A value of **TRUE** for the variable valid indicates that a calibration was done before, and valid values are contained in the other variables. If this is the case, the value of valid is then changed to **FALSE** (Line 5), and the values contained in the variables wakeupintervalvalue and sendintervalvalue (also stored in the NVRAM) are read (Line 4) and relayed to the application-level code through signalling the event `calibrationfinished` (Line 25). A valid value of **FALSE** indicates that the node did not come from a restart, and there are no valid values stored in the NVRAM. In such a case, protocol operation then proceeds to PHASE 2.

2. **PHASE 2.** PHASE 2 begins by starting the radio (Line 8). After the radio is started, the variable `current_wakeupinterval`, which contains the current wake-up interval value being tested, is set to the value of the lower bound, specified when `calibratewakeupinterval` command was called (Line 9). A full set of packets is then sent, separated by the specified send interval (Lines 12-15). The number of packets in a set (the window size) is a protocol parameter which will be tested later on to study its effect on protocol effectiveness. After a set is successfully sent, the value of the variable valid in the NVRAM is set to **TRUE** (Lines 17-18), and the values of the just-tested wake-up interval and send interval are written to the NVRAM (Line 16). The variable `current_wakeupinterval` is then incremented by the value of resolution (also specified when `calibratewakeupinterval` command was called) (Line 19), and a new set is sent (Lines 10, 20).

   PHASE 2 has two possible outcomes. In the first outcome, the node restarts in the middle of sending a set, since the wake-up interval is

no longer supportable. Provided that the application code will call the `calibratewakeupinterval` command upon bootup, the maximum receiver wake-up interval value that is supportable can then be retrieved from the NVRAM by PHASE 1 of the protocol. In the second outcome, all the values specified in the `calibratewakeupinterval` lower bound-upper bound range turn out to be supportable, and the node does not shutdown or restart. In this case, protocol operation proceeds to PHASE 3.

3. **PHASE 3.** PHASE 3 begins by setting the value of the NVRAM variable valid to **FALSE** (Lines 21-22). The radio is then turned off (Line 23), and the event `calibrationfinished` is signalled (Line 25). Note that the event `calibrationfinished` can now possibly be signalled by two conditions: from PHASE 1, wherein it will contain the measured maximum receiver wake-up interval, and from PHASE 3, wherein it simply indicates that the entire range of values specified in the `calibratewakeupinterval` command is supportable. To differentiate between the two cases, the event `calibrationfinished` includes the variable frompreviously which is **TRUE** when signalled by first phase of the protocol, and **FALSE** when signalled by the third phase.

2.5.5 Mote-autonomous protocol: Results

We test the mote-autonomous protocol using a node running an application wired to EHCalibrateSELFC. The application code is designed to immediately call the `getpendingcalibration` command upon booting. The command can cause the node to restart, after which the next call to `getpendingcalibration` command will result in the signalling of the event `calibrationfinished` with a valid calibration result. Alternatively, the node may *not* restart and the event `calibrationfinished` would still be signalled, but with an invalid calibration result. In this case, the application will call `getpendingcalibration` again, but

Algorithm 1 Algorithm for mote-autonomous protocol

```
1: Inputs: sendintervalvalue; lowerbound; upperbound; windowsize; resolution
2: get copy of variable valid from NVRAM                    ▷ start of Phase 1
3: if valid == TRUE then
4:     get copy of variables wakeupintervalvalue and sendintervalvalue from NVRAM
5:     valid ← FALSE
6:     update variable valid in NVRAM                       ▷ end of Phase 1
7: else                                                     ▷ start of Phase 2
8:     start radio
9:     wakeupintervalvalue ← lowerbound
10:    while wakeupintervalvalue ≤ upperbound do
11:        sentmsgs ← 0
12:        while sentmsgs < windowsize do
13:            send packet
14:            sentmsgs ← sentmsgs + 1
15:        end while
16:        update variables wakeupintervalvalue and sendintervalvalue in NVRAM
17:        valid ← TRUE
18:        update variable valid in NVRAM
19:        wakeupintervalvalue ← wakeupintervalvalue + resolution
20:    end while                                            ▷ end of Phase 2
21:    valid ← FALSE                                        ▷ start of Phase 3
22:    update variable valid in NVRAM
23:    turn off radio
24: end if
25: signal calibrationfinished   ▷ end of Phase 3, called in Phase 1 if flow does not proceed to Phase 2
```

with a higher range of wake-up interval values to test. The process will repeat itself until a range is found which will cause the node to restart.

After getting a valid calibration result, the node will then proceed to use the result in sending packets to a base station. The packets that it sends to the base station also contain the calibration result value. This way, we are able to monitor what the actual calibration result is, and whether it is effective (as will be evidenced by the client being able to repeatedly send packets to the base station for a continuous period of time).

We test three capacitor values: 3000 μF, 4000 μF, and 5000 μF, and utilized 30 seconds as the send interval. We use a resolution of 50 ms, and set the lower bound and upper bound values as 0 ms and 1000 ms, respectively. An

important parameter of the protocol is the measurement window size. We test four measurement window sizes: 2, 4, 6, and 8.

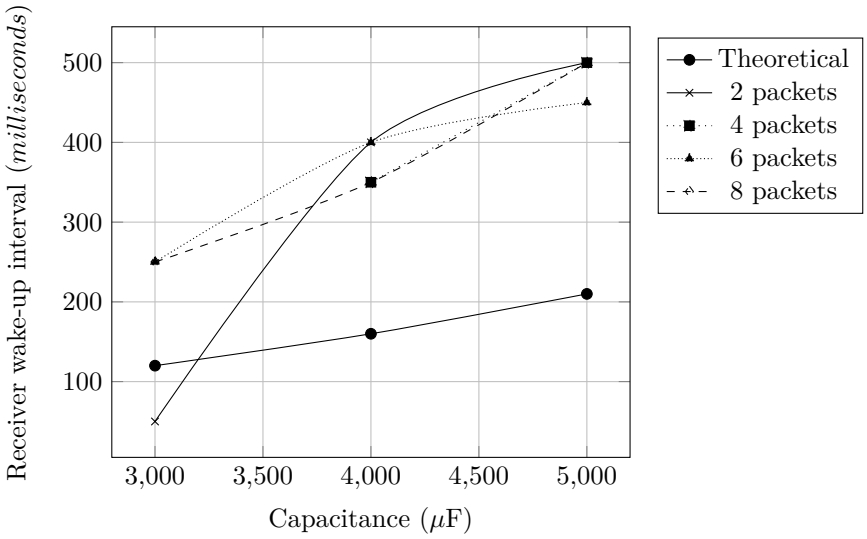


Figure 2.7: Maximum supportable wake-up interval values: theoretical vs experimental (measured by calibration protocol).

The results given by the calibration protocol, and the values given by the computational method, are plotted in Figure 2.7. From Figure 2.7, it is apparent that in all capacitor values tested, the actual receiver wake-up interval that can actually be supported by the sender is much greater than the value given by Equation 2.3. Figure 2.7 also shows that the window sizes tested mostly gave very consistent limit values, differing only by a single resolution step. That is generally the case except for the lowest capacitor value and small window sizes tested. For instance, for the window size of 2 and the capacitor value of 3000 μF, the measured receiver wake-up interval is only 50 ms, lower than the theoretical value, and lower than that given by the other setups. For a window size of 4 and the capacitor value of 3000 μF, the calibration process failed. Inspecting the voltage and current readings from our datalogger, we find that this phenomenon is due to the power consumption of the NVRAM writing process. The process of writing to the NVRAM is energy-intensive, even more so than the packet sending process at short receiver wake-up intervals. To illustrate, a section of

the supply voltage readings while running an experiment is plotted in Figure 2.8. Figure 2.8 shows the readings during the last send of a set (labelled A) and the NVRAM write which follows it (labelled B). The NVRAM write actually consumes more energy than the send itself. When the capacitor value is small, the amount of charge contained in the capacitor is easily depleted. When the window size is also small, the NVRAM writes are more closely spaced than when the window sizes are bigger. Thus, the system is not given enough time to charge the capacitor before the next NVRAM write. This causes the node to shutdown during the NVRAM write process itself or during the succeeding set. Thus, while the window size does not matter much when the capacitor values are large, it is preferable to use larger window sizes for smaller capacitor values.

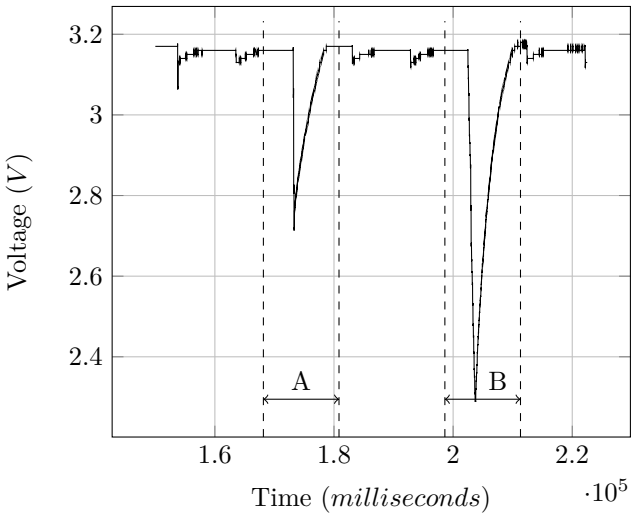


Figure 2.8: Segment of voltage readings.

Nevertheless, it must be noted that in general the smallest functional window size possible should be used, as larger window sizes translate to longer calibration processes, and thus longer disruptions to normal node or network operation. The times taken by the experiment setups to finish are plotted in Figure 2.9. Note that the times plotted in Figure 2.9 are greater than the actual calibration times; they are actually the time elapsed between the connection of the node to the energy harvesting system and the reception of the base station of the

first post-calibration packet from the node. Nevertheless, the application-level code component is fairly constant across different setups, thus the differences between the times plotted in Figure 2.9 are caused by the calibration process (protocol level component).

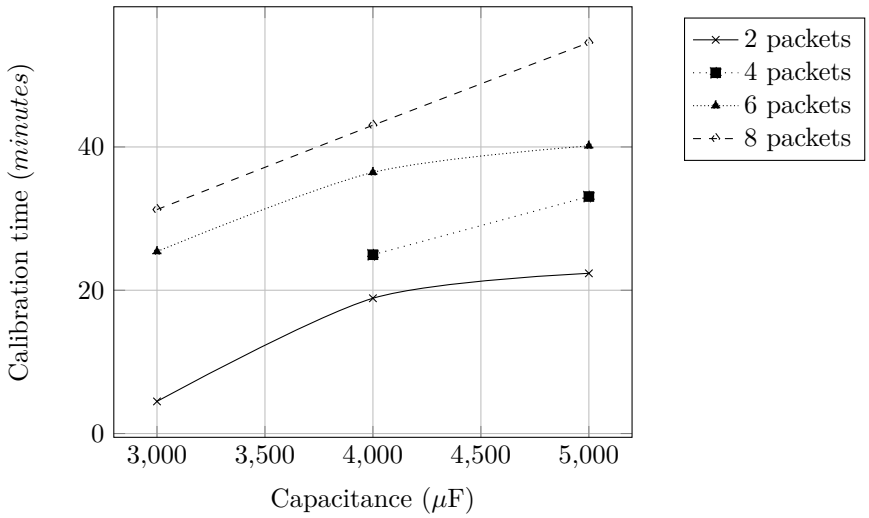


Figure 2.9: Experiment times for different setups.

2.5.6 Correctness of algorithms

The algorithms work and are correct since the since the shutdown is caused by the sender draining charge from the boost capacitor for a duration (equivalent to the receiver wake-up interval) longer than what the boost capacitor can supply, given the size of the boost capacitor and the length of charge time. Provided that the sender will keep on increasing its assumed receiver wake-up interval, it will eventually come across a current draw duration that will induce it to shutdown and restart. The crucial part is the how the sender will know post-restart the value at which it restarted. This is where the two protocols differ. In the client-server architecture-based protocol, the server listens in and remembers the last wake-up interval being tested by the sender. It then tells the sender this value upon request after the sender restarts. In the mote-autonomous protocol the sender mote itself remembers the wake-up interval value by writing it to the

non-volatile memory, and accessing the value after restarting.

We note that neither protocol will cause the participating nodes to be trapped in a lockdown or uncertain state, as long that the assumptions regarding the links are met - specifically, the requirement that the links are *bidirectional*. As previously mentioned, as long as the sender is set to repeatedly try high receiver wake-up intervals as long as it has not shutdown,the limit of the boost capacitor will eventually be reached, inducing a shutdown, and causing the protocol to proceed. As for the receiver (server) in the client-server architecture-based protocol, there are moments where it has to wait for certain messages from the sender (client) - for example, waiting for the *ACCEPTANCE* message in Phase 1, or waiting from a *CALIBRATION* message from a sender (client) that has already shutdown. However, these waits were coded to have corresponding *timeout mechanisms*, ensuring that the receiver (server) will not wait indefinitely. Should the timeout occur before the reception of the message being waited for, the node proceeds to normal operation and terminates the protocol.

2.5.7 Related work

In a general sense, our work is related to *power management* algorithms for energy harvesting systems. Examples of such work include [162] and [134]. Such studies generally deal with the optimization of packet sending schedules and packet sending rates. However, the approaches used in such work are usually highly analytical in nature and rarely empirical. Thus, the results are rarely directly (or immediately) applicable to real-world setups, as the abstractions or idealizations needed to do such analyses leave out or misrepresent many aspects of real systems.

More relevant to our work are the energy-efficient or energy-aware Medium Access Control (MAC) protocols designed for WSNs. A significant number of such work have been carried out through the years, and the reader will do well to consult papers such as [34] for a more comprehensive discussion of the body

of work available. Examples of such work include [62][46][54][61]. Such work are usually very specific to a certain system - for example, [62] is specific to systems that are powered through RF energy transfer - and thus not applicable to our setup. Even those which are designed for general EH systems were usually tested through simulations - that is, the protocol itself does not exist yet at a form that can already be deployed. While simulations are arguably more realistic than pure analyses, simulations still miss out on many aspects of real world deployments. Moreover, some of the algorithms and protocols presented in such papers may not even be implementable because of technical reasons (limited node resources, for example) [45].

For these reasons, the closest work to ours are probably those of MAC protocols that were *actually-implemented* and designed for *energy efficiency*. Such protocols are not necessarily designed for energy-harvesting systems, but operate efficiently enough (or can be configured to do so) so that system operation while energy harvesting is possible.

It must be emphasized that we are *not* proposing a new MAC protocol, but rather, we are proposing a way of how the default and widely-used TinyOS MAC protocol or radio duty cycling scheme, when used with energy storage devices like the Cymbet EnerChip, can be *tailor-fitted* for better performance. Admittedly, a lot of MAC protocols incorporating improvements over TinyOS' LPL in terms of energy efficiency are already available - these include LPL's counterpart in ContikiOS [29], ContikiMAC [28], WiseMAC [31], and X-MAC [16]. All share LPL's basic philosophy, but improves upon it by more precise packet transmission timing (in the case of ContikiMAC), keeping a schedule of possible receiver wake-up times (in the case of WiseMAC) and the use of a packetizing radio (in the case of X-MAC). Using a slightly different philosophy (receiver-initiated protocol instead of sender-initiated), we have [121] and [30]. Nevertheless, given the large existing userbase of TinyOS and the popularity of LPL, we believe that our protocol can be of great use to the wider WSN community, particularly to those who intend to deploy functional energy-harvesting

WSNs and want to use a well-documented and widely-used MAC protocol.

2.5.8 Summary

If maximum performance and utility are to be had from energy-harvesting WSNs that utilize energy storage devices such as the EnerChip, it is imperative that the maximum receiver wake-up interval that is supportable by the sender be accurately determined. Our experiments show that computational methods give overly conservative values. Manual experimentation is accurate but labour-intensive. In this subchapter we present two protocols which allow nodes to autonomously determine the said limit: one utilizes a client-server architecture, while the other allows the nodes to derive the said limit mote-autonomously, even without the help of other nodes. The protocols are shown to be effective and robust, although for the mote-autonomous protocol the window size parameter (number of sends used in the tests) must be carefully chosen when the boost capacitor value is small. The protocols can be easily integrated into any user-level application node.

Aside from saving man-hours, the protocols also have the additional advantage of being usable while the network is already deployed *in situ*. Capacitors have a limited lifespan and their capacitance change as they age; therefore, the maximum supportable receiver wake-up interval of a sender also changes over time. Without these protocols, nodes can possibly stop working as pre-programmed values no longer fit the physical parameters they were set for. With the regular use of the protocols, the nodes themselves can track such changes and adjust the program parameters without any human intervention, ensuring continued system functionality.

2.6 Methods for computing charge time in multiple-pulse load applications

In this subchapter, we extend the equations and methods given in Chapter 2.4 to accommodate multiple current draws of varying kinds. We also discuss ways of shortening the charge time (mainly by using an empirically-derived alternative to the analytically-derived one), thus possibly improving the performance of a system (as more current draws can be done within a given period of time).

For systems with only a single pulse component or only one component which consumes a high level of current for an extended period of time, Equation 2.1 and Equation 2.2 will suffice in deriving the size of the capacitor needed. Equation 2.5 will then give the duty cycle of the system. For applications that have two or more components that have significant current draws (or what we call *multiple-pulse load applications*), however, Equation 2.1 and Equation 2.2 will not suffice.

In the future, multiple-pulse loads will become more and more common in wireless sensor network nodes as the sensors that are attached to motes grow, not just in number but also in sophistication (for example, [87]). These multiple-pulse load applications pose no problem for battery-powered systems since batteries can supply high levels of current for long periods of time. For energy harvesting-powered systems that use energy storage devices like the EnerChip however, such multiple-pulse load applications will require a revision of Equation 2.1 and Equation 2.5 or the creation of new techniques that will ensure continuous system operation. In summary, the problem is this: *given multiple pulse loads, executed in succession, with possibly differing levels of current draws and draw length durations, what should be the size of the boost capacitor, and what should the intervals (charge times) be between the pulse loads?*

In this subchapter, we consider the case wherein there are two pulse loads in the system. Nevertheless, all of the methods can be easily and readily extended to accommodate three or more pulse loads.

References will be made to *small* and *large* pulse loads. By this we mean with respect to the size of the boost capacitors computed for each pulse load using the method described earlier in Chapter 2.4. *Pulse loads* will also be referred to simply as *loads*. Once again, since all capacitors involved in this subsection are boost capacitors, all references to 'capacitors' should be taken to mean 'boost capacitors'. Figure 2.10 shows a diagram of the current draw of a system with only a single type of load (small load). Figure 2.11 on the other hand shows current draw of a system which only executes large loads. The current draw times and charge times for both systems are assumed to be derived using the method presented earlier.

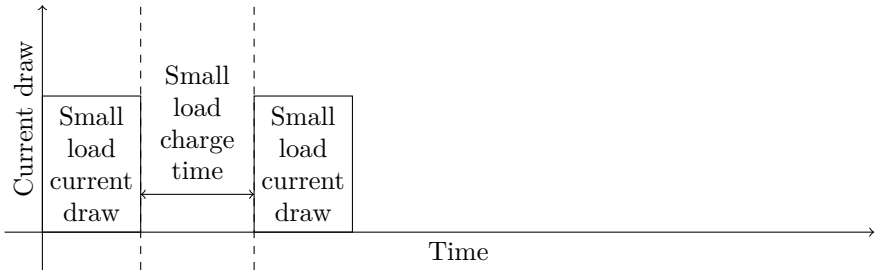


Figure 2.10: Diagram of current draw for a system with only a single type of load (small load).

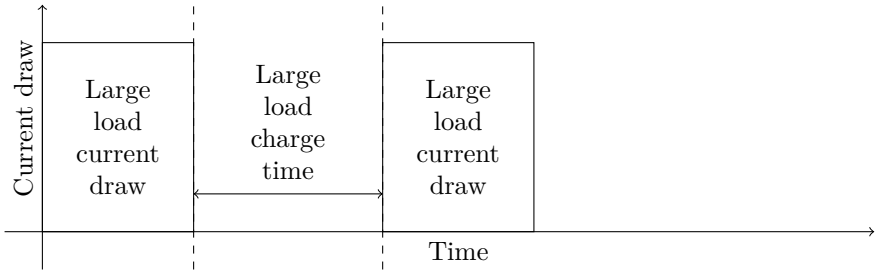


Figure 2.11: Diagram of current draw for a system with only a single type of load (large load).

For all the methods described, the capacitor of the larger load is utilized (which easily answers the first part of the problem posed above). The larger capacitor is utilized because it will accommodate both current draws. The capacitor of the small load can possibly prove insufficient for the current draw

of the large load, completely depleting it or dropping its voltage below the minimum level required for system operation.

2.6.1 Methods for computing charge time in multiple-pulse load applications

Lazy method

In the simplest of the four methods, the computationally-derived charge time for the larger of the two loads (the one needing a larger capacitor) is simply used for both loads. The method is guaranteed to work: since the charge time is enough to recharge the capacitor after a large current draw, it will definitely suffice for the smaller current draw. The main advantage of the method is its simplicity, while its main disadvantage is the performance loss that comes with it. The charge time of the larger draw is usually larger than that of the smaller draw's original charge time - thus, the overall duty cycle of the system will be lower than what it will be if the two draws were simply concatenated. Since the smaller draw does not deplete the capacitor as much as the larger draw, it follows that the smaller draw can actually make do with a shorter charge time - something which will be taken advantage of by another method. Figure 2.12 shows a diagram of the current draw of a system with alternating small and large loads, with the charge times determined with the Lazy method.

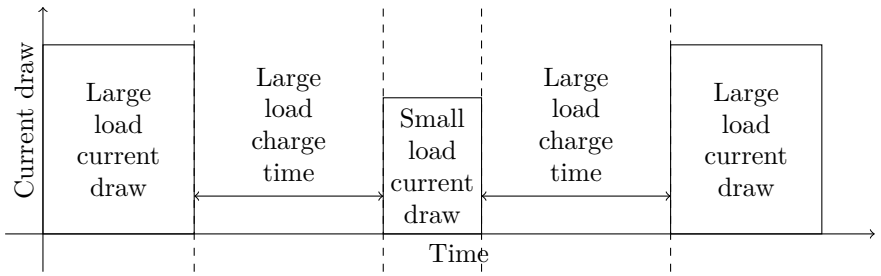


Figure 2.12: Diagram of current draw for a system with two types of loads, with charge times determined by the Lazy method.

Concatenation method

In this method, the two or more loads are simply *concatenated*, or executed in sequence, each retaining its original, analytically-derived charge time. The reason this method can work is primarily because of the inherent conservativeness of the analytical method, and indeed, preliminary experiments have shown that this method can work for certain combinations of loads. The main advantage of this method is its simplicity: the computations necessary are exactly the same as that of the single-load system, only performed several more times. The disadvantage of this method is that it does not work for all load combinations: when the loads are vastly different in magnitude, the capacitor may not have enough time to be charged at a sufficiently high enough voltage level before the next load manifests itself. Figure 2.13 shows a diagram of the current draw of a system with alternating small and large loads, with the charge times determined with the Concatenation method.

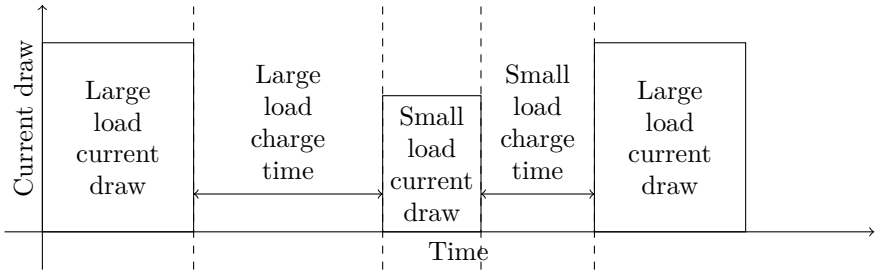


Figure 2.13: Diagram of current draw for a system with two types of loads, with charge times determined by the Concatenation method.

Aggressive analytical method

In this method, the analytical method is optimized to reflect the fact that while there are now two different loads in the system, there is only one capacitor. The value of the large load charge time will remain the same - although it will now become the charge time *after* the execution of the large load and *before* the execution of the small load. This is because the capacitor's recovery time after a large load should remain the same as before - it is the large load's optimal

capacitor size that is being used in the system after all. What will change and be newly computed by this method is the charge time *after* the small load and *before* the large load. To compute this:

1. Denote the $\mathsf{t_{Draw}}$ and $\mathsf{R_{Load}}$ of the *small* load as $\mathsf{t_{DrawSmall}}$ and $\mathsf{R_{LoadSmall}}$, respectively, and denote the optimal capacitor size for the large load as C, compute $\mathsf{V_{MinSmall}}$ with

$$\mathsf{V_{MinSmall}} = \frac{\mathsf{V_{Max}}}{e^{\frac{\mathsf{t_{DrawSmall}}}{\mathsf{R_{LoadSmall}} \times C}}} \tag{2.6}$$

   This is, in effect, the level at which the voltage across the capacitor will drop to when the small load makes its current draw.

2. The $\mathsf{t_{Charge}}$ for the small load (denoted as $\mathsf{t_{ChargeSmall}}$) can then be computed from Equation 2.3, using $\mathsf{V_{MinSmall}}$ instead of $\mathsf{V_{Min}}$.

Figure 2.14 shows a diagram of the current draw of a system with alternating small and large loads, with the charge times determined with the Aggressive analytical method.

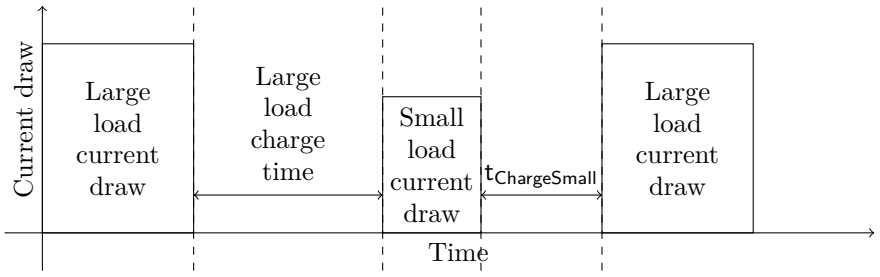


Figure 2.14: Diagram of current draw for a system with two types of loads, with charge times determined by the Aggressive analytical method.

Opportunistic method

All methods discussed so far focus on finding charge times that will enable multiple-pulse load applications to function in an energy harvesting setting. They have the advantage of ensuring predictability and uniformity in operation,

as it is known exactly when the loads will be executed, and at what intervals. There may be situations however, when predictability and regularity are not strictly required - one can, for instance, be more concerned about getting as many sensor readings as possible than getting such readings at perfectly defined intervals. In such situations, all of the previous methods are disadvantaged in their being conservative. This is also demonstrated in Chapter 2.5 (Figure 2.3). Analytical methods are inherently conservative since they are designed to ensure system operation across many conditions. For instance, the rate at which power is supplied by the energy harvesting system actually varies from one time to another: it is a function of the current light intensity applied to the solar panels as well as the storage system's state of charge. Nevertheless, when carrying out computations for the analytical method, the value that must be assumed is the minimum supply rate, or at most, the average. Thus, there are moments where the system can actually support better performance (i.e., higher duty cycles). When predictability and regularity are not strictly required, such extra performance not being utilized can be considered as *wasted* performance.

To avoid such waste whenever possible, we propose a new method for charge time determination. Instead of utilizing predetermined charge times for a load, an *opportunistic* charge time is instead used: the system executes a load whenever possible, subject to the charge level of the capacitor. For this, we rely on the MSP430's supply voltage supervisor (SVS), which also has counterparts in other microcontrollers. The SVS, when activated, continuously monitors the level of the supply voltage of the microcontroller, and sets the SVSOP bit of the 8-bit SVSCTL register to **1** whenever it goes below a user-predefined value. The enabling and disabling of the SVS, as well as the setting of the voltage threshold, are likewise done by setting certain bits in the SVSCTL register. To implement the method, the execution of the load must be gated or wrapped by another code which activates the SVS and checks the SVSCTL register. The pseudocode for such wrapping or gating, called `Gated_Exec`, is outlined in Algorithm 2.

The code is parameterized by two values: volt_limit and thresh_count.

Algorithm 2 Wrapping/gating code for Opportunistic Algorithm (`Gated_Exec`)

```
1: Inputs: volt_limit, thresh_count
2: threshold_exceeded = FALSE
3: threshold_exceeded_counter = 0
4: low_power_flag = 0
5: while threshold_exceeded_counter < thresh_count do
6:     Delay(1s)
7:     Enable SVS monitoring, define low voltage as volt_limit
8:     Delay(10ms)                          ▷ Allow SVS monitor to settle
9:     From SVS register, set low_power_flag value
10:    Disable SVS monitoring
11:    if low_power_flag ≠ 1 then
12:        threshold_exceeded = FALSE
13:    else
14:        threshold_exceeded = TRUE
15:    end if
16:    if threshold_exceeded is TRUE then
17:        threshold_exceeded_counter = threshold_exceeded_counter + 1
18:        threshold_exceeded = FALSE
19:    else
20:        threshold_exceeded_counter = 0
21:    end if
22: end while
23: Execute Load
```

Lines 5-22 comprise the main *while* loop of the code which prevents the execution of the load while conditions are not yet met, primary of which is the threshold_exceeded_counter reaching thresh_count. Line 6 imposes a 1-s delay between loop iterations. Line 7 enables the SVS and sets the voltage threshold (defined by the user through volt_limit). Line 8 gives time for the SVS to settle. In Line 9, the SVSOP bit of the SVSCTL register is checked, and its value copied to low_power_flag. Line 10 disables the SVS between loop iterations, to save power. Lines 11-15 check the value of the low_power_flag, effectively converting it to a boolean variable threshold_exceeded. A value of **TRUE** for threshold_exceeded indicates that the supply voltage has exceeded that of the threshold. Lines 16-21 deal with determining the new value of threshold_exceeded_counter: it is incremented by 1 if the threshold is exceeded (threshold_exceeded is **TRUE**) and reset to 0 otherwise. The purpose of the parameter thresh_count (to which threshold_exceeded_counter is compared) is

to allow users to define a minimum amount of time by which the threshold voltage level must be exceeded *continuously* before considering the capacitor as sufficiently charged for load execution. This is important since the relationship between capacitor voltage and its state of charge is not linear - the rise of capacitor voltage slows down during the latter stages of the charging process. Thus, even if the threshold is set to a high value, it will not be prudent to immediately consider the capacitor as sufficiently charged just because its voltage was observed to surpass the said threshold - this is especially true for capacitors of significant size. Another use for this parameter is for rate control in routing protocols such as directed diffusion [52]. In directed diffusion, a node on the path to the sink may request upstream nodes to slow the generation of data or packets because it can no longer handle the packet volume. The execution of the actual load happens in Line 23, which will only be reached upon escaping the loop embodied in Lines 5-22.

2.6.2 Testing the methods for computing charge time in multiple-pulse load applications

To test the methods, we create a dual-pulse load application using LPL. The mote alternates between two kinds of sends, the first (Load 1) sends to a receiver with a wake-up interval of 0.26 s, and the second (Load 2) sends to a receiver with wake-up interval of either 0.63, 1.02, or 1.42 s, making for a total of three setups. The sends are configured as broadcasts, so the node upon sending will attempt the send for as long as the amount of time specified by the receiver wake-up interval - thus, the length of the receiver wake-up interval is equivalent to the sender's current draw length for that load. In contrast, when doing acknowledgement-enabled unicasts, the sender will stop the send operation upon receiving an acknowledgement from the receiver, to save energy. The analytically-derived capacitor sizes and charge times for the four loads (Load 1, Load 2A, Load 2B, and Load 2C) when each is considered in isolation are tabulated in Table 2.9.

Table 2.9: Load table

	Draw length (s)	Capacitor size (μF)	Charge time (s)
Load 1	0.26	2,000	29.18
Load 2A	0.63	5,000	70.7
Load 2B	1.02	8,000	114.5
Load 2C	1.42	11,000	157

To test that the setups are working, all the packets are received by the base station, which indicates packet reception through a terminal printout. The system is powered by energy harvesting during the experiments, with the Cymbet EnerChip fully charged beforehand. In each setup, the capacitor utilized is that of the larger load: for example, in the Load 1-Load 2A setup, the capacitor used is 5,000 μF. Each experiment is run for a period long enough so that 25 packets from each type of send are sent by the mote.

All methods are tested on all three setups and found to be functional. For the Opportunistic method, thresh_count is set to 1, and volt_limit is set to 3.35 V, for both loads. The application running on the mote is slightly modified for the experiment used to test the Opportunistic method - to enable us to keep track of the charge times, the length of the charge time preceding a send is sent within the packet as payload. The length of the charge time can be determined by the number of iterations of the loop embodied in Lines 5-22 of Algorithm 2. To ensure that the system is in continuous operation (not restarting), each packet also contains a sequence number that is regularly incremented after a send operation.

For the Load 1-Load 2B setup, Figure 2.15 plots the opportunistically-determined charge times for both loads, along with the charge times determined by the Concatenation method.

It can be seen in Figure 2.15 that the opportunistically-derived charge times vary across time, indicative of the constantly changing charge rate of the energy-harvesting system. It must be noted that in the opportunistically-derived pair, the charge time for the large load is always smaller than that of the small load, in contrast with the analytically-derived pair. To understand this, one must

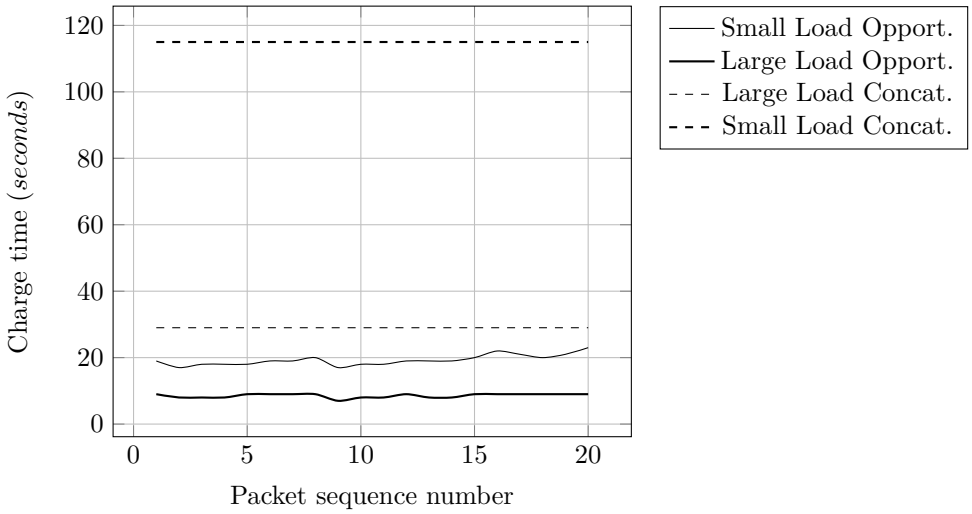


Figure 2.15: Charge times: Opportunistic method vs Concatenation method.

remember our definition of charge time: it is the period of time before a current draw (or active period). A small load is preceded by the large load, which draws a significant amount of current from the capacitor. Thus, charging the capacitor to an acceptable voltage level will take time. By the same token, the large load is preceded by the small load, which relatively does not draw much current. Therefore, it may be more prudent to compare the opportunistically-derived large load charge time with the analytically-derived small load charge time, and vice versa. Even when such comparisons are made however, the opportunistically-derived values are still significantly smaller than the analytically-derived values. For instance, the average opportunistically-derived charge time for the small load is 19.25 s, and the analytically-derived charge time for the large load is 114.5 s.

The duty cycles associated with each method for the three setups are normalized against that of the Concatenation method and plotted in Figure 2.16 and Figure 2.17. Figure 2.16 plots the duty cycles associated with the three analytical methods, while Figure 2.17 plots the average duty cycle measured when the Opportunistic method is used. The data for the Opportunistic method is plotted separately since its numbers are significantly larger than that of the

other three.

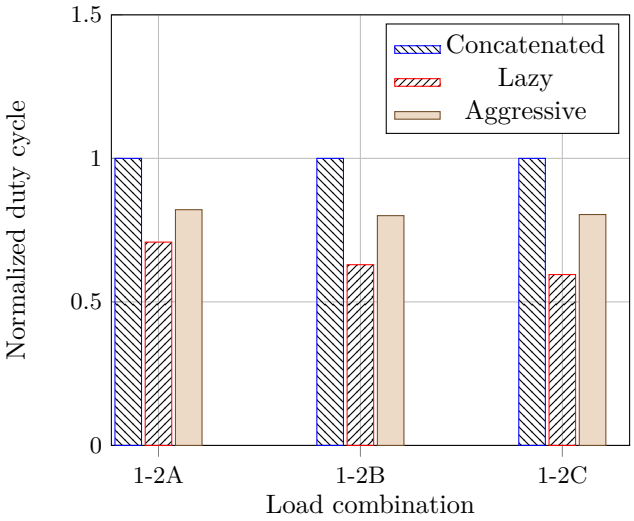


Figure 2.16: Normalized duty cycles, analytical methods.

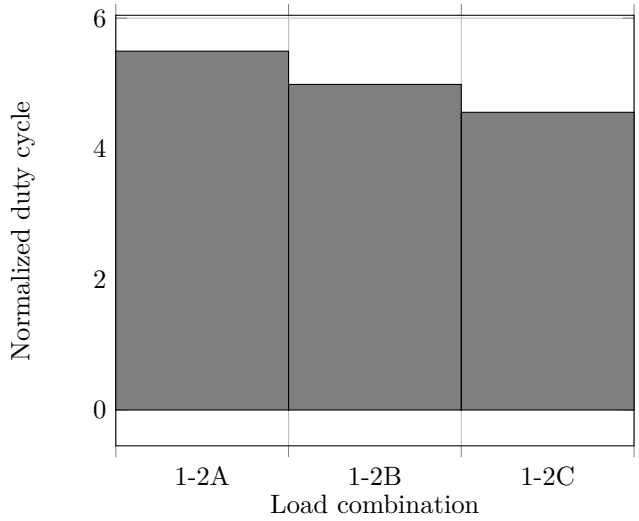


Figure 2.17: Normalized duty cycles, Opportunistic method.

Figure 2.16 shows the Lazy method and the Aggressive Analytical method being inferior to the Concatenation method. This is to be expected. As already mentioned, the Lazy method is guaranteed to work but has the drawback of performance loss, which grows proportionately with the disparity between the sizes of the small load and the large load. This too, is apparent when the results

are compared across setups. The Aggressive Analytical method performs better than the Lazy method, but still worse than the Concatenation method. The Aggressive Analytical method is also guaranteed to work, but even with the other charge time tailor-fitted for the voltage drop caused by the small load, the value is still larger than that of its counterpart in the Concatenation method. The voltage level from which the charge period has to recover is higher (because the capacitor is larger), but the capacitor is now also more difficult to charge.

Figure 2.17 shows normalized data for the Opportunistic method. The Opportunistic method performs significantly better than all three other methods, with the lowest in the three setups showing a 350% improvement over the Concatenation method. These results signify that if the requirements of perfect data regularity and predictability can be relaxed, significantly more performance can be had from the system.

2.6.3 Summary

We present and discuss several methods for computing charge times for energy-harvesting nodes that run multiple-pulse load applications. We propose and test an opportunistic method for use in cases where the requirement of predictability and regularity in data generation can be relaxed. Our method is shown to exhibit 350%-400% improvement over analytical methods when it comes to the generated duty cycle. While the Opportunistic method is presented in the context of multiple-pulse load applications, it can readily be applied to single-pulse load applications.

It must be noted that the methods and protocols presented in Chapter 2.5 and this subchapter are *not* mutually exclusive. They actually work on different parts of an operation cycle. The protocols from Chapter 2.5 ensure that receiver wake-up interval is as long as it can be, resulting in energy savings for the receiver. This is actually done by *maximizing* the current draw time at the sender. The methods presented (specifically the Opportunistic method), on the other hand, *minimizes* the charge time. A situation where both can be applied

is when an energy-harvesting WSN node wants to send data that will not fit in a single packet. The maximum receiver wake-up interval that can be supported can be derived using the protocols in Chapter 2.5. The receiver can set its wake-up interval to this value, maximizing its energy savings. The sender can then opportunistically derive the charge time for each send, potentially resulting in the multi-packet data getting sent faster.

CHAPTER 3

Technical requirements: Energy-harvesting noise-sensing WSNs and the noise metric required by noise codes

3.1 Overview

In this chapter, we examine how currently-prevailing technical requirements affect the design, performance, and applicability of energy-harvesting WSNs. Specifically, we consider the noise measurement metric specified in many of the currently-enforced noise codes (such as the European Union's Environmental Noise Directive [33] and the New York City Noise Code [95]), and how it affects the applicability of low power or energy-harvesting WSNs to the problem of noise pollution sensing. We empirically derive or demonstrate the limitations of energy-harvesting WSN nodes as they carry out the noise code-compliant noise sensing process and based on these limitations propose an alternative design or architecture.

3.2 Introduction

As discussed in Chapter 1.1, WSNs can potentially help with noise measurement, as they enable the simultaneous and continuous gathering of data over wide geographic regions. This is in contrast to sound level meters (SLMs), which are usually only deployed when investigating a noise-related complaint. Most modern cellular phones have sensitive microphones, and this capability, when coupled with the ability to geolocate (through GPS readings) and connect to the Internet, opens up the possibility of 'crowdsourcing' the noise measurement process. Nevertheless, WSNs still hold an advantage over crowdsourced noise

measurements. In contrast to cellular phone readings contributed by users, WSNs can provide data that is uniform (since the measurement devices will be of the same type) and properly contextualized (since each node's location and orientation will be known). Such data can help in building noise maps, a goal shared by many 'smart city' initiatives all over the world (for example, [77]). However, the high energy requirement of the noise measurement process exacerbates the difficulties associated with maintaining a WSN deployment, since batteries will have to be frequently replaced. Powering noise-sensing WSNs with energy harvesting can help alleviate or solve this problem.

Several noise-sensing WSN nodes have been demonstrated before [130][43][21]. Most of these studies utilize the same basic design for their nodes, with the differences between them mainly in the networking algorithms and the node (microcontroller) platform: that is, they sample and record the waveform, and from the data, compute *continuous equivalent A-weighted sound level*, or $\mathsf{L_{eqT}}$. This is primarily due to $\mathsf{L_{eqT}}$ being the primary metric on which many of the noise codes are based. None of the previous studies attempt to power their system through energy harvesting.

Given the benefits that can be had from energy-harvesting noise-sensing WSNs, it is important to determine whether they can be realised or implemented, the level of performance that can be reasonably expected from them, and their most likely limitations. To this end, we perform empirical analysis on an implementation of the 'traditional' design while it is powered by energy harvesting. Our analysis reveals that noise sensing *can* be carried out in energy-harvesting WSNs - however, because of the high energy requirements of the noise measurement process, and the limited capabilities of low power WSN nodes (especially with regard to the memory available), energy-harvesting noise-sensing WSNs (at least as they are at present) will have difficulties meeting the requirements specified in many of the currently-enforced noise codes. If some requirements (specifically the requirement that the output noise metric be $\mathsf{L_{eqT}}$) can be relaxed, an alternative design can be created that is more amenable to

being powered by energy harvesting.

Noise codes intend to regulate *noise* pollution. However, what will be discussed in the following sections mainly concern *sound* loudness and *sound* pressure levels. Technically, noise can simply be defined as 'unwanted sounds'. What qualifies as noise is highly subjective: what is noise for one person may not be considered as such by another. A sound need *not* be loud to be considered noise; however, there is widespread consensus among experts that very loud sounds have adverse effects on human health and well-being [74]. As such, governments (through noise codes) focus on this aspect of noise.

It must be noted however that 'loudness' itself is a psychological and subjective measure. Technically, loudness is the 'psychological correlate' of the physical strength (amplitude) of a sound [4]. A sound's amplitude can be objectively defined with measures such as sound pressure levels (SPL, or simply *sound levels*), although it does *not* solely define how human beings *perceive* the strength of a sound. Human perception of sound strength is also affected by the sound's duration, bandwidth, and comprising frequencies. In this study, consistent with most noise codes, we only consider the 'sound pressure' aspect of a sound's loudness (it must be noted however that most noise codes also take into account how human beings perceive different frequencies differently by requiring that readings be A-weighted).

In the following discussions, we will sometimes take the liberty of referring to a sound's SPL as its 'loudness'. In *most* cases, the relationship *will* hold - a sound with higher SPL will be perceived by a human being as being 'stronger' (hence louder) than one with lower SPL. Nevertheless, we emphasize that the usage of the word in this context is somewhat loose and imprecise.

The remainder of the chapter is organized as follows. The next subchapter (Chapter 3.3) discusses the 'traditional' sound level measurement process as carried out by WSN nodes. This is followed in Chapter 3.4 by an empirical analysis of a 'traditional' design implementation that is powered by energy harvesting. Chapter 3.4 also surveys the capabilities and limitations of the said combina-

tion. Chapter 3.4 provides the backdrop and motivation for the design decisions made in Chapter 3.5. Chapter 3.5 presents an alternative to the usual node design or architecture, which, while non-compliant with existing noise codes, is better-suited to being powered by energy harvesting.

3.3 Sound level measurement in WSNs

Sound is a mechanical wave which uses air as a medium. As the wave travels, it induces pressure fluctuations which are then detected by the human ear, or in mechanical/electronic devices, a transducer. Greater energy in the wave translates to bigger pressure fluctuations, which humans then experience as the *loudness* of a sound. The human ear is an extremely sensitive device: the difference between the smallest and biggest pressures that the human ear can sense, vary by 12-13 orders of magnitude. Representing such a huge range can be cumbersome in the linear scale, so sound pressure level is usually defined in a logarithmic scale, with the unit of *decibels* [113] [14].

$$\mathtt{L_p(t)} = 10 \times \log_{10}(\frac{\mathtt{p^2_{RMS}(t)}}{p^2_{ref}})(dB) \tag{3.1}$$

In Equation 3.1, $\mathtt{L_p(t)}$ is the instantaneous *sound pressure level* (SPL) of a sound, while $\mathtt{p_{RMS}(t)}$ is the root mean square (RMS) of the pressure. p_{ref} is the standard reference pressure set at 20 μPa. 20 μPa is conventionally set as the minimum pressure detectable by the human ear.

$\mathtt{p_{RMS}(t)}$ is not truly instantaneous since the RMS has to be computed over a period of time. While the length of time over which the RMS must be computed is not standard, it must at least be equal to or longer than the period of the lowest frequency being measured. In a discrete-time system, assuming that `p(i)` is the pressure measured at the sampling instance i and N is the number

of samples over which we are making the RMS computation, we have

$$\mathsf{p}_{\mathsf{RMS}} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}\mathsf{p}^2(i)} \tag{3.2}$$

While the length of time for the RMS computation can be adjusted, what is usually used in measuring noise levels over extended periods of time is $\mathsf{L}_{\mathsf{eqT}}$, or the *time-averaged sound level*. $\mathsf{L}_{\mathsf{eqT}}$ is taken from the average of successive $\mathsf{p}_{\mathsf{RMS}}$ values. Again, assuming a discrete-time system and N as the number of $\mathsf{p}_{\mathsf{RMS}}$ values taken over the time for which $\mathsf{L}_{\mathsf{eqT}}$ is being computed, we have Equation 3.3 [113]:

$$\mathsf{L}_{\mathsf{eqT}} = 10 \times \log_{10}\left(\frac{1}{N}\sum_{i=0}^{N-1}\frac{{\mathsf{p}_{\mathsf{RMS}}}^2(i)}{p_{ref}^2}\right)(dB) \tag{3.3}$$

For the microcontroller of the node to be able to calculate Equation 3.2, p should be in a form that is digital in nature: the pressure waveform therefore has to go through transformations. Firstly, there is the microphone/transducer, which senses the pressure fluctuations in the air using a thin membrane. The physical fluctuation of the membrane induces voltage fluctuations. The voltage fluctuation induced by the pressure fluctuation is defined by the *microphone sensitivity*, denoted by S in Equation 3.4 [7]:

$$\mathsf{S} = 20 \times \log_{10}\left(\frac{\mathsf{E} \times p_0}{E_{ref} \times \mathsf{p}}\right)(dB) \tag{3.4}$$

E_{ref} is conventionally set to 1 V while p_0 is set to 1 Pa. E is the resulting voltage swing when pressure changes by p. With S defined (through a datasheet, for example), E can be easily derived from Equation 3.4.

The output of the microphone is a continuous voltage waveform. In most noise-measuring systems, the microphone output is also *A-weighted*, meaning its frequency components are attenuated or amplified to match how the human ear perceives different frequencies (since the human ear does not perceive fre-

quencies equally [36]). The voltage waveform is also usually preamplified using an operational amplifier (op-amp)-based active amplifier before being processed by the microcontroller's analog-to-digital converter (ADC). The ADC samples the voltage waveform in discrete time steps and discretizes the voltage level into binary integers. The output of the ADC is a stream of binary integers, which are then stored in the microcontroller's memory. Two parameters define the operation of the ADC: the sampling rate and the word width. A higher sampling rate translates to a more faithful representation of the original signal. The word width is the number of bits available for representing the sampled value. For instance, if the word width is only two bits, the ADC will only be able to differentiate between four levels (since $2^2 = 4$). If the continuous voltage values vary between 0 and 1 V, values between 0 and 0.25 V will be encoded as 00 and values higher than 0.25 V but no higher than 0.5 V will be encoded as 01, etc.

The output of the ADC can already be processed by the microcontroller's CPU and used as input to Equation 3.2. While Equation 3.2 requires the sound pressure level, the ADC output will suffice as an input. Barring the precision loss introduced by the ADC, the relationship between the integer value and the original pressure level is linear: the output can be scaled later. Alternatively, the programmer can also choose to convert the ADC output into its Pa equivalent before having the CPU do the computation. The downsides of this approach are the extra computational steps and the need for floating point numbers, which are not supported by all microcontroller platforms.

3.4 Noise-sensing in WSNs: Survey of capabilities and limitations

In this subchapter, we evaluate the challenges related to measuring sound levels on a WSN node augmented with Commercial-Off-The-Shelf (COTS) or 'almost-COTS' components. The design we use in this work is not necessarily optimal or better than those used in previous studies. Instead of being optimized or

specialized, we believe that our system is representative of a *generic* noise-pollution monitoring WSN node design - because of this, the lessons learned from our experiments are easily applicable to other designs. Aside from evaluating node-level capabilities and limitations, we also evaluate the prospect of having the system powered by energy harvesting.

It must be noted that in our experiments, we focus on capabilities and limitations not covered or discussed by previous studies on noise-sensing WSNs. In particular, we focus on aspects that are related to the prospect of having the system powered by energy harvesting. As such, we focus less on often-discussed and well-explored metrics, such as the accuracy of the dB readings produced by noise-sensing WSN nodes.

3.4.1 Methodology and physical setup

The node platform we utilize is the TelosB [123], running TinyOS.

For the energy harvesting component of our setup we utilize the CBC-EVAL-09 [23].

For sound-sensing, we use the electret microphone breakout board (Model BOB-09964) manufactured by Sparkfun [102]. The output of the microphone is preamplified by an inverting amplifier based on the OPA344 operational amplifier [152], with the gain permanently set to 100. The microphone model in the breakout board has a sensitivity (S) of -46 dB, and *Signal-to-Noise Ratio (SNR)* of 58 dB. The microcontroller of the TelosB, the TI MSP430, has several 12-bit ADC channels. The microphone's output is connected to one of these ADC channels.

Our sound pressure level measurements are taken using a professionally-calibrated Casella CEL-240 sound level meter (SLM) [75], while our voltage and current readings are taken with a PicoLog1216 datalogger [99].

The algorithm that we utilize in computing the loudness of a sound is presented in Algorithm 3. The algorithm takes the output of the ADC, stores them in array readings (Line 4) and computes the RMS (Lines 7-10). It must be noted

that 2048 must be subtracted from each element of the array readings (Line 8) because of the $\frac{V_{CC}}{2}$ offset in the output of the microphone. Algorithm 3 also specifies the possible entities which can perform a specific computation step (node or base station, Lines 3, 5, and 12). This is in recognition of the fact that many of the steps need not be done on the node itself. For instance, the node can just take the ADC samples, transmit them to the base station, and have the base station do the remainder of the processing. Since the base station usually possesses computational power that is orders of magnitude higher than that of the node, it will be able to perform the computations faster than the node can. Nevertheless, offloading all the computations to the base station may not be a practical solution since it will entail transmitting all the ADC samples from the node to the base station. Depending on the sampling rate and the time window considered, the ADC may produce hundreds to thousands of samples. The packet size utilized by the node is limited in size, thus transmitting all samples will take several packets. The transmission of such packets is time-consuming and energy-intensive, thus processing the samples locally will usually end up being the more cost-effective option for the node.

Algorithm 3 Algorithm for RMS computation

```
1:  Inputs: arraysize
2:  Output: total
3:                                                          ▷ node:
4:  take arraysize ADC samples using DMA, store in array readings
5:                                                  ▷ node/basestation:
6:  processed ← 0
7:  while processed < arraysize do
8:      accumulator ← accumulator + (readings[processed] − 2048)^2
9:      processed ← processed + 1
10: end while
11: subtotal ← accumulator / (processed+1)
12:                                                         ▷ basestation:
13: total ← √subtotal
```

3.4.2 Experiments and results

Electret microphone board capabilities

In the first experiment, we test the capabilities of the electret microphone board. The board is powered by a variable power supply set to 3 V. We use a variable power supply instead of the energy harvesting system, since in our preliminary experiments, the energy harvesting power supply tends to become unstable when directly connected to the microphone board. This is because the energy harvesting power supply utilizes interfacing capacitors, and duty cycling has to be carried out to ensure that the capacitors are regularly charged from the EnerChip batteries. Such a function can be carried out by the node, but not by the microphone board alone. The datalogger records the output of the microphone.

The experiment consists of exposing the microphone to increasing sound levels to detect the maximum SPL that it can detect. A sound level meter situated near the microphone enables us to monitor the SPL of the sound the microphone is exposed to. Figure 3.1 plots the readings from the microphone board as it is exposed to a 1000 Hz tone of increasing volume.

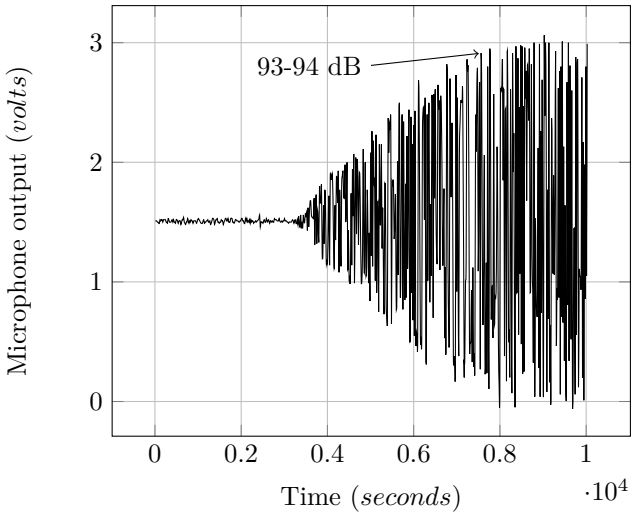


Figure 3.1: Microphone output range test, supply = 3 V.

As indicated in the Figure 3.1, the microphone board produces voltage

swings that increase in amplitude along with the tone volume up until the 93-94 dB range. Beyond this level, the voltage swing no longer increases in amplitude. The voltage swing at 93-94 dB and beyond is already equivalent to the supply voltage level. Thus, the microphone electret board, when powered by 3 V, can only differentiate between sound pressure levels that are lower than 93 dB.

93 dB as an upper bound does not suffice for a noise pollution monitoring system, especially one that aims to help in the enforcement of noise codes such as the European Union's Environmental Noise Directive [33]. Nevertheless, there are solutions to such limitations. It must be noted that the main source of the 93 dB limit is not the microphone itself, but the *preamplifier* which amplifies the signal coming from the microphone. The preamplifier in the electret breakout board is an op-amp-based inverting amplifier. The output of op-amp-based amplifiers can only swing between the levels of its supply voltages V+ and V- (also called *rails*). For the op-amp in the microphone board, V- is 0 V (GND) and V+ is 3 V (in our case). It is not that the microphone output can no longer swing higher when the sound pressure level is higher than 93 dB; rather, the microphone signal, post-amplification, ends up being higher than 3 V, and thus can no longer be properly produced by the amplifier (when this happens, the output is said to have *clipped*). The simplest way therefore to extend the microphone board range beyond 93 dB is to *lower* the gain of the preamplifier. Doing so will decrease the output resolution of the microphone board, but it will allow the system to extend the range of its detectable or differentiable SPLs. A rather ingenious solution to the same problem is actually implemented in [43], where the preamplifier is designed to have two modes, a high gain mode and a low gain mode. The mode is digitally selectable, and the switching is decided and carried out by the node microcontroller. The switching decision is made depending on the ADC readings: once clipping is detected in the ADC readings, the gain is decreased; when the ADC readings are below a certain threshold, the gain is increased. It must be noted however, that the upper bound cannot be indefinitely increased by simply decreasing the preamplifier gain. While the

microphone board upper bound is currently limited by the preamplifier gain, the microphone itself has a theoretical maximum input SPL specified in the datasheet (110 dB).

The SPL upper bound detectable by the microphone can also be analytically derived. This is done by

1. dividing the rail width (3 V in our case) by the product of the microphone sensitivity (S) and the preamplifier gain
2. using the quotient from the previous step as input to Equation 3.1

The SPL upper bound as a function of the preamplifier gain is plotted in Figure 3.2. The theoretical upper bound for our current preamplifier gain is 109.5 dB, almost a match to the theoretical absolute limit imposed by the microphone. We attribute the disparity between our experimentally-derived limit and the analytically-derived limit to the deviation of the microphone from the datasheet specifications, and the fact that the microphone has a non-zero output even when it is detecting no sound (this is demonstrated later).

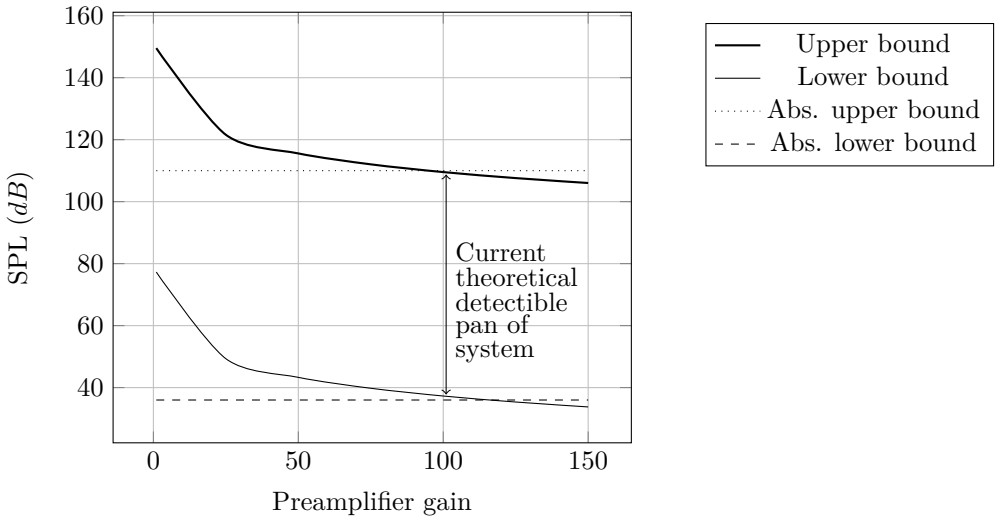


Figure 3.2: Upper bound and lower bound of detectable SPL values as a function of preamplifier gain.

Microcontroller ADC capabilities

In the second experiment we test the capability of the microcontroller's 12-bit ADC. The experiment consists of having the ADC sample the microphone board output and then sending the readings to a desktop computer. The ADC is set to sample at 10,000 Hz, using direct memory access (DMA). In Algorithm 3, this corresponds to having the node ADC do just the sampling and having the base station do the remainder of the processing. There is, however, the significant difference that in this experiment, the node transmits the packets to the base station via a USB cable rather than via radio, as the case will be in an actual implementation. We also do not make the base station perform the RMS computation, instead simply using it to collate the readings so that we can see whether the ADC is performing well enough for us to be able reconstruct the signal.

The first limitation we come across while doing the experiment has nothing to do with the ADC per se, but with the memory size of the microcontroller. We utilize DMA in getting the readings; essentially, we pass an array to the ADC via a TinyOS command. The array is then returned to the user application containing the readings. Due to the node being limited in memory, the largest array that we can create for the readings only has around 4000 elements (each of which is a 16-bit word) in size. This means that the longest reading that we can get at any one time will only consist of 4000 samples: at a sampling rate of 10,000 Hz, this translates to 0.4 seconds. After 0.4 seconds, the readings will have to be processed, which will cause a gap in the sample stream.

The second limitation is the settling time required by the ADC when it is first turned on: our readings indicate that at the current setting that we employ, the first readings of the ADC are erroneous - of the first 4000 readings, the first 3000 readings are unstable and inaccurate (Figure 3.3).

To see whether the original signal is recoverable from the ADC output, we expose the microphone to tones of different volumes (again, a 1000 Hz tone is used), and then plot the readings produced by the ADC. To make sure that the

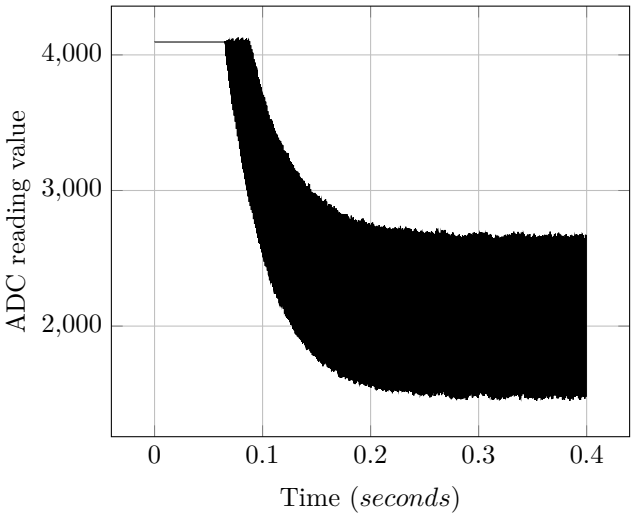


Figure 3.3: Plot of ADC readings.

ADC has settled, we only utilize the last 1000 of the 4000 readings. The plots are presented in Figure 3.4.

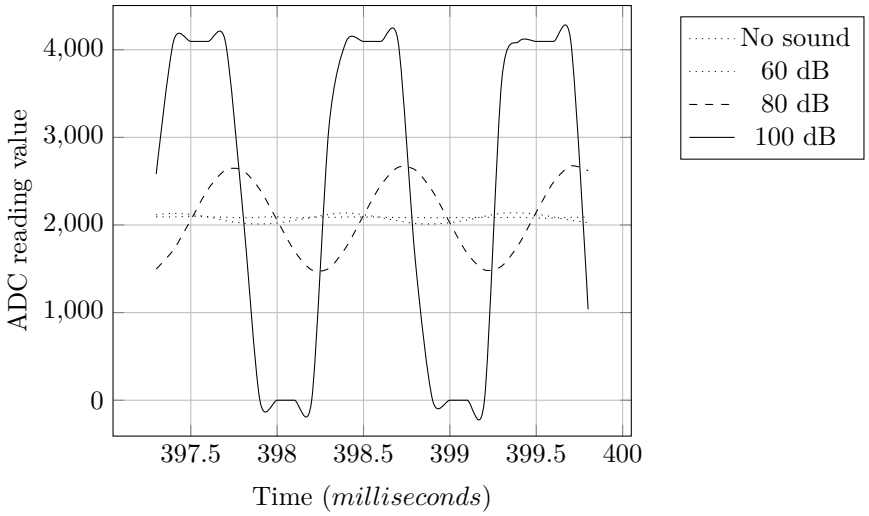


Figure 3.4: Plot of ADC readings, different dB levels.

Figure 3.4 shows that from the output of the ADC, the original signal is actually reconstructible, and the microphone-ADC block can differentiate between different acoustic dB levels. The waveforms are successfully reconstructed for all dB levels except that of 100 dB, where there are continuous readings of

values 4095 and 0. This is not a limitation of the ADC per se, but that of the microphone (as previously discussed). The ADC and the microphone share common supply voltages and thus, GND is mapped by the ADC to 0 while 3 V is mapped to 4095, the highest unsigned integer value representable by 12 bits.

Microcontroller computational capabilities

In our third experiment, we test the computational capabilities of the node. We have the node perform all the steps in Algorithm 3 which the node can perform, and have it send the result to the base station via radio. The system is powered through energy harvesting, although the experiments are carried out inside the laboratory and the light for the solar panels is supplied by a lamp.

Every 90 seconds, the application running on the node takes samples, does the computations, and transmits the computed value to the base station. The ADC is run at a sampling rate of 10,000 Hz, and only the last 1000 samples are utilized in each reading. For the radio transmissions, the node and the base station employ duty cycling via LPL. The base station is set to have a wake-up interval of 0, meaning it is always awake. Thus, the radio of the node is turned off most of the time, and consumes energy for a very brief period of time when transmitting.

The computation results sent by the node to the base station while exposed to sounds of different sound levels are plotted in Figure 3.5. Note that we take four readings at each sound level.

As can be seen in Figure 3.5, the output of the node computation tracks the sounds' SPL. It must be noted that Figure 3.5 shows that the output tracks the SPL even at the 100 dB level - that is, it still shows increased readings. This is interesting since the two previous experiments have demonstrated that the system can only accurately discern values up to the 93-94 dB level. This can be explained by taking into account in detail how the computational steps which the node carries out to produce the output. When the SPL exceeds 93-94 dB, the output of the microphone is clipped, and this in turn leads to consecutive

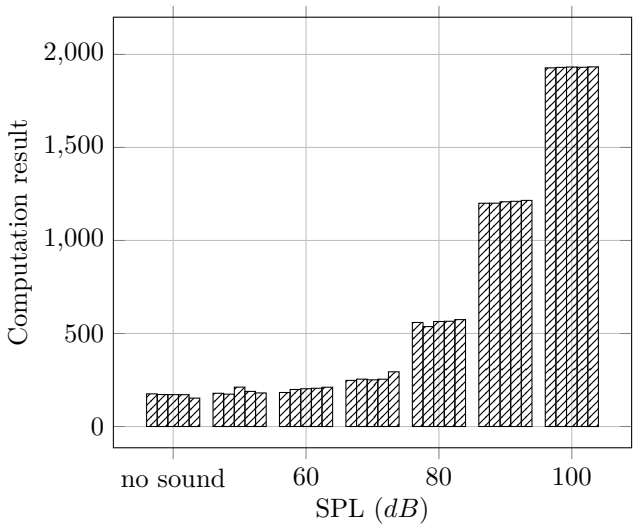


Figure 3.5: Node-computed RMS, energy harvesting-powered.

ADC readings of 0 and 4095 (Figure 3.4, 100 dB). As the SPL gets higher and higher, the clipped regions increase, and so does the number of consecutive 4095 readings. This, in turn, leads to a higher sum, which translates to a higher computation output.

There is another limitation which is not demonstrated by our experiment, but is apparent in the system specifications. Each array element is produced by the 12-bit ADC, and is stored in a 16-bit variable. The largest output of the subtraction-squaring operation (Algorithm 3, Line 8) occurs when the output of the ADC is at the extremes, 0 or 4095 - which will give 4,194,304 (or 4,190,209). This will no longer fit in a 16-bit variable, which can only store up to 65,535. We therefore store the result directly in a 32-bit accumulator. The largest unsigned integer which a 32-bit word can store is 4,294,967,295. Dividing 4,294,967,295 by 4,194,304 gives 1023. This means that to remove the possibility of overflow, the number of elements being summed should not exceed 1023. Of course, the case where the all 1023 elements have the value 4,194,304 will only occur when the microphone output is constantly clipped. In most cases, more than 1023 elements can probably be summed without inducing overflow, even if there is some clipping in the microphone output (such as in Figure 3.4, 100 dB).

However, in the absence of any other analytical method for deriving an upper bound for the 'safe' number of elements which can be summed, we deem 1023 as a conservative limit worth following.

Figure 3.5 also reveals another limitation of the system, this time regarding the minimum SPL which can be detected.

It is apparent from Figure 3.5 that even when no sound is fed to the microphone, it still produces an output. This is due to the inherent noisiness of the microphone, defined by its Signal-to-Noise Ratio, or SNR. All microphones have a noise floor, also called Equivalent Input Noise (EIN). Below the EIN level, the microphone cannot differentiate between an input sound and its own internal noise. Thus, for a sound to be detected by a microphone, its SPL must be above that of the microphone's EIN. The EIN can be derived from the SNR by simply subtracting the SNR from 94 dB (assuming that the SNR is conventionally measured). The microphone that we use has an SNR of 58 dB, and therefore has an EIN of 36 dB.

Nevertheless, while the EIN defines the lower bound SPL for the microphone, the overall system lower bound is affected by other components such as the preamplifier gain, and the width of the ADC. The effect of the preamplifier gain has already been discussed. The ADC width affects the lower bound because for a change in voltage to be detected by the ADC, it must change by an increment high enough to be detected by the ADC. The minimum increment is defined by the width of the ADC. In our case, since the supply is 3 V and the width is 12-bits, the minimum change in voltage level must be 0.0007324218 V (3 V/4096). It is therefore not enough for a sound to go beyond the EIN of 36 dB, it must also cause the voltage output to change by at least 0.0007324218 V to be detectable. The minimum dB level above the EIN required for a sound to be detectable can be computed by

1. dividing 0.0007324218 V by the product of the microphone sensitivity (S) and the preamplifier gain

2. computing the Pa equivalent of the EIN via Equation 3.1

3. using the quotient from first step as input to Equation 3.1, but with the value from the previous step as reference (denominator) value instead of p_{ref}

The SPL lower bound as a function of the preamplifier gain is also plotted in Figure 3.2. At the preamplifier gain value that we use (100), the system lower bound is 1.3 dB higher than the EIN, or 37.3 dB. It is apparent from Figure 3.2 that increasing the preamplifier gain improves the system SPL lower bound: however, the gain can not be indefinitely increased, since values below 36 dB are no longer valid. The same way that the microphone input SPL limit defines the absolute upper bound, the microphone EIN defines the absolute lower bound.

There are, however, two other ways of getting as near to the EIN as possible aside from adjusting the preamplifier gain. Firstly, a more sensitive microphone can be used (higher S). Secondly, a wider ADC can be used. A wider ADC with the same supply voltage will require a voltage increment smaller than 0.0007324218 V. Both are not feasible options for us in this work, but they are things worth considering for someone designing a similar system from scratch.

Energy harvesting

In our final experiment, we investigate how the system will perform when powered by energy harvesting. The setup in the previous experiment is already powered by energy harvesting, but in a controlled environment (i.e., inside the laboratory). For this experiment, we deploy the node outdoors and have it powered by actual sunlight. For this we utilize a custom-built weatherproof test rig for outdoor testing solar-powered WSN nodes, nicknamed the *Ice Cream Tub*. An internal view of the *Ice Cream Tub* is shown in Figure 3.6, while an external view is shown in Figure 3.7.

The experiment is run for 24 hours, from 09:00 of June 22, 2013, to 09:00 of

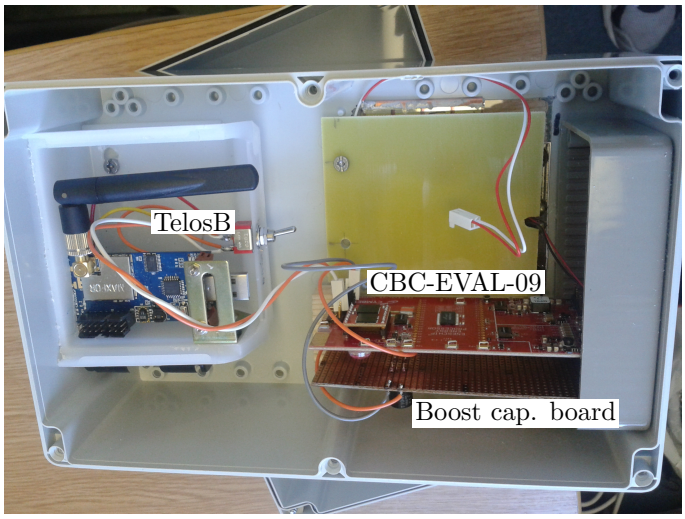


Figure 3.6: Ice Cream Tub (microphone not attached), internal view.

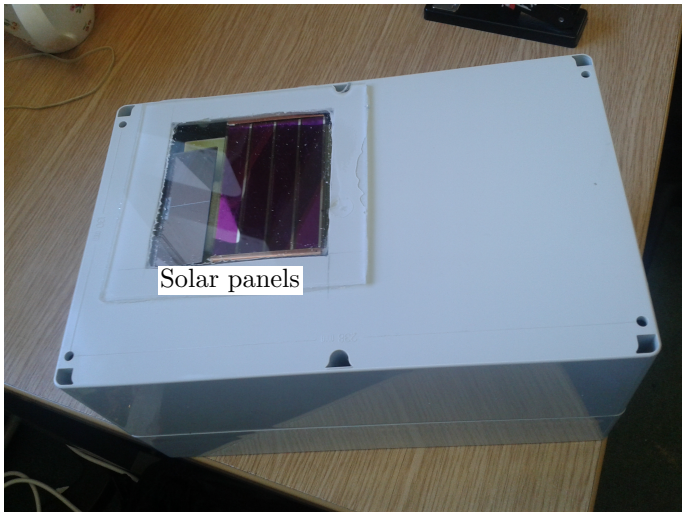


Figure 3.7: Ice Cream Tub, external view.

June 23, 2013. The weather for those days is mainly sunny with slight showers during the midday of June 22. The node transmitted its results to a base station located roughly 150 m away, in a slightly higher elevation.

Before deployment, the energy storage device of our system is first fully charged by exposing the solar panels to 1300 lx+ light for an hour inside the laboratory - 55 minutes is roughly the exposure time needed for the system to reach full charge according to the datasheet [23]. We use the same application

as in the previous experiment.

To see how the system is affected by environmental factors, another node (which we call the *observer* node) is deployed inside the *Ice Cream Tub*. The observer node is powered by batteries, and reports the readings of its onboard Hamamatsu S1087 light sensor [84] to the base station. The observer node is positioned beside the energy harvesting-powered node's solar panel, so the readings made by the observer node closely reflects the intensity of the light which the solar panel receives. The readings of the oberver node are also wirelessly transmitted to the base station.

The light intensity recorded by the observer node throughout the day, and the number of packets received by the base station from the solar-powered node, are plotted in Figure 3.8. It must be noted that we also have the actual RMS readings from the solar-powered node, but do not consider it relevant for two reasons: firstly, our deployment area is generally quiet, and secondly, the *Ice Cream Tub* does not have any port hole for the microphone (so it is inside the enclosure for the entire experiment).

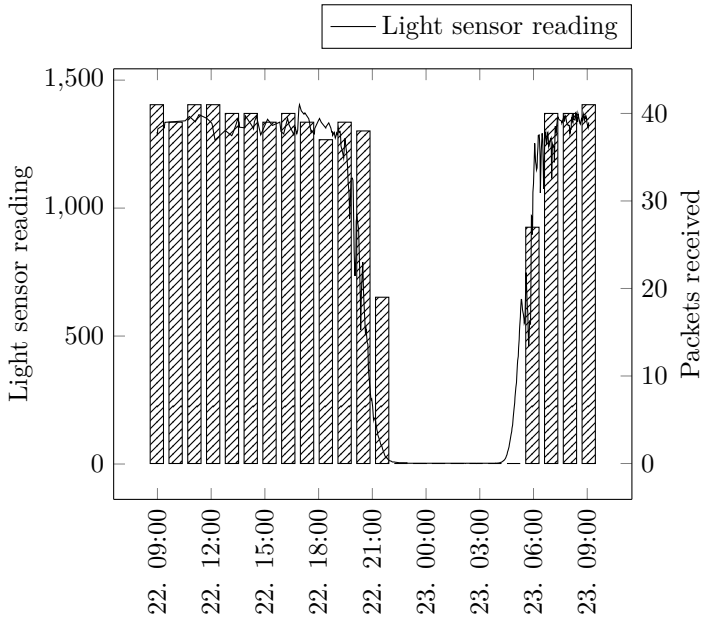


Figure 3.8: Light sensor readings from *observer* node, number of packets received from solar-powered node.

The first observable feature of the light sensor readings in Figure 3.8 is how well it reflects a typical British day in mid-June. The long daylight hours are apparent (the experiment is held just a day after the summer solstice) - as the light sensor's readings only zeroed roughly between 21:00 of June 22 and 04:00 of June 23. It is also apparent from Figure 3.8 that the performance of the system, at least when it comes to generating and sending packets, is closely tied to the amount of sunlight available. Despite the energy storage device being fully charged before the deployment, the number of packets generated and sent by the node drops down to zero in the middle of the night. An interesting feature of Figure 3.8 is the lag between the two plots - a number of packets are still successfully sent by the node to the base station roughly an hour after sunset; likewise, it took some time after sunrise before the transmissions resume. The first represents the system purely living off the energy stored in the energy storage device; the second represents the system charging up to a certain level first before being able to power the node.

The observed behaviour is to be expected, as the energy storage device we utilized is extremely limited in size (only 100 μAh). Even if the node receives enough surplus energy, it does not have enough capacity to store it. We verify that this is the case with our energy storage device by fully charging the EVAL-09 indoors (without load) and then removing the solar panel. The EVAL-09 is then connected to the node, which is not able to remain operational for 7 hours. In such cases, simply increasing the energy storage capacity will enable the node to continue operating through the night.

It must be noted however that not all shutdowns can be attributed to the insufficient capacity of the energy storage device. It is entirely possible that the energy storage device can store enough energy for night time operation, but not enough energy is harvested during the daytime. In such cases, the power consumption of the node must be decreased so that the harvested energy suffices for continuous operation. Controlling the power consumption of a node in response to the harvested energy from the environment is facilitated by the

power management algorithm. Adjusting a node's power consumption can be carried out in several ways, including duty cycling, changing sensor sensitivity, etc. The options available to a power management algorithm depend both on the node's hardware and the software it is running. Power management algorithms are further discussed in Chapter 3.5.2.

3.4.3 Summary

We successfully demonstrate that measuring sound loudness is possible with an energy harvesting-powered WSN. Nevertheless, the system has several system limitations imposed by different components at different parts of the RMS computation process. These limitations, along with their causes and possible solutions, are tabulated in Table 3.1.

We believe that the greatest limitation of the current system is not specified in Table 3.1: the inability to continuously take in readings. The limitation is brought about by the confluence of several factors specified in Table 3.1: Firstly, there is the limited memory which requires the program to process the readings every 4000 samples; Secondly, there is the limited number of array elements which can be summed with the assurance that no overflow will occur; Finally, there is the need for energy harvesting-powered systems to duty cycle. It is a serious limitation especially when considered in light of what most noise codes require. For example, the European Union's Environmental Noise Directive [33] requires perfectly continuous readings as it, in essence, sets its standards by RMS averages taken over an entire day or night. In comparison, the system tested in Section 3.4.2 is only able to take in readings constinuously for 0.4 seconds.

Despite the shortcomings of the current system (which includes its non-implementation of A-weighting), the fact that it can measure sound loudness while being powered by energy harvesting is already promising. The system is assembled from 'almost-COTS' components, and implements absolutely no power management. With customization (for instance, in the areas of preampli-

fier gain control, storage, and power management algorithm), we are certain that the range and capabilities of the system can improve. Nevertheless, it canot be denied that significant challenges have to be overcome before energy-harvesting noise-sensing WSNs can attain performance sufficient enough for them to be useful in the implementation of currently-existing noise codes.

Table 3.1: Summary of limitations uncovered in the empirical analysis

Limitation	Cause	Possible solution(s)
Limited range, only up to 93-94 dB	Microphone board output clips	Dynamically adjustable preamplifier gain
First 3000 samples unusable	ADC settling time	Lower sampling rate, better microcontroller
Only 1000 samples can be summed	Limited word width	Better microcontroller
Sound must be louder than 37.3 dB to be detectable	Microphone characteristic, limited ADC width	Better microphone or better microcontroller
Generation and transmission ceases during the night	Not enough capacity for energy received, or not enough energy received	Larger storage capacity and/or power management

3.5 Energy-harvesting noise sensing WSN node: An alternative design

In this subchapter, we present a node extension design which will enable a node to detect noise while being powered by energy harvesting, specifically, solar energy harvesting. The design offers several advantages over the design tested in Chapter 3.4, overcoming some of the limitations surveyed in the empirical analysis carried out in the previous subchapter. While the design does *not* conform with the standards of measure utilized by any of the existing noise codes, such a network can still be immensely useful in many urban settings. In terms of administration, our design makes for a relatively low-cost and low-maintenance system, something that may not be currently achievable with WSNs that are designed to comply with existing noise codes.

3.5.1 Design

Design rationale

While most of the previous work that involve sound sensing has done so by recording the sound waveform [130][43][21], we opt for an approach that is centered on a peak detector, for the reasons cited above. This means that our system does not produce SPL values, but the peak level of sound recorded within a period of time. Technically, what is recorded by the system is the peak *voltage amplitude* induced by the microphone (and preamplifier) within a period of time. However, the voltage swing produced by the microphone is proportional to the sound SPL, so it is representative of the sound level. Such information is insufficient for the requirements of most existing noise codes, but as discussed, such information, combined with an energy-harvesting or (non-rechargeable) battery-free operation feature, will suit noise sensing in many urban settings.

Our decision to adapt a peak detector-based design is also motivated by the limited capabilities of the TelosB [123] - in Chapter 3.4, we demonstrate how

at 10,000 Hz sampling rate, the limited memory space of the MSP430 [51][151] microcontroller only enables continuous sampling of up to 0.4 s. In comparison, for many systems that monitor $\mathsf{L_{eqT}}$, $\mathsf{p_{RMS}}$ is usually computed over 1-s intervals [74]. It must be noted that $\mathsf{L_{eqT}}$ can still be computed despite this limitation because the 1-s sampling interval is not standard, and the ADC can be simply be activated again for another set of readings. There will potentially be a gap between the sets of readings (which will be a function of the microcontroller processing speed), but even that can be minimized by parallelizing processing and sampling using direct memory access (DMA). Even with DMA however, two problems still remain. Firstly, the need to compute the RMS of the samples every 0.4 s will consume a lot of energy over time. Secondly, and more significantly, the number of such readings that can be accumulated will be severely constrained by the amount of space - it must be noted that space is one of the factors that contributed to the 0.4 s limit in the first place. This can be alleviated by shorter sampling windows (i.e., shorter sampling sequences) or by frequently sending data to the sink. The former, however, will negatively affect the accuracy of the readings, while the latter will be costly in terms of energy.

In comparison, a peak detector-based system does not record the waveform at all and only generates a digital reading at the end of a sampling period. A significantly larger number of readings can then be stored, resulting in greater flexibility in how and when such readings are processed. For example, assuming that time bounds requirements are met, readings can be accumulated so that several are sent to the base station in a single packet. The timing of sending can also be dynamically set to coincide with periods of high energy generation, helping to keep the system in energy-neutral operation. Alternatively, the packet sent can be externally triggered, with the data pulled out of the node by a mobile sink or data mule. Such options are simply very limited or not available in systems that are severely constrained in the amount of data that they can store.

It must be noted that a peak detector-based design does not necessarily

consume less power than a design which carries out wave sampling. The system unfortunately still has to duty cycle, meaning it cannot be in the active state all of the time. Ways of alleviating this limitation will be discussed later. While the ADC is no longer continuously active and sampling the microphone at a high frequency, a new sub-circuit is added to the system, one which continuously runs during the active part of the operation cycle. Our measurements show that the power consumption of the sub-circuit is at par with (or just slightly smaller than that of) an ADC at a high-frequency sampling operation. Nevertheless, the flexibilities afforded by a peak detector-based system gives significantly more options for power management algorithms which can lead to energy savings in the long run.

Basic design

For our work, we use the ultra-low-power wireless sensor module ('mote') TelosB [123], described in Chapter 2.3. For the energy-harvesting component of our setup, we utilize the CBC-EVAL-09 [23], also previously described in Chapter 2.3.

For the sensor design, we utilize an ADMP401 MEMS microphone [6]. Compared to conventional electret microphones, MEMS microphones offer the advantage of minimal size, lower power usage, and a better signal-to-noise ratio (SNR) [8]. The microphone output is preamplified by an op-amp-based inverting amplifier with a gain of 100. The preamplified output is then fed into a peak detector circuit with a load-isolated storage capacitor. This peak detector design is more suited for long hold times than the simpler single op-amp-based alternative. The details of the peak detector design are discussed in detail in [38][20][138]. The output of the peak detector is connected to ADC channel 0 of the TelosB, where it is sampled at the end of a sensing period. All three operational amplifiers (op-amps) in the design are OPA344s [152].

To facilitate duty cycling, the supply lines of the microphone, the preamplifier, and the peak detector are gated by the high-side switch ADP194 [5]. The

ADP194 is digitally controlled by the TelosB using a digital I/O pin. During the sensing period, the *Enable* pin of the ADP194 is set to **HIGH** by the TelosB, enabling the current to flow to the microphone, the preamplifier, and the peak detector.

The design also features a MAX323 [92] digital switch, which provides a digitally switched line between the storage capacitor and a discharge resistor (whose other end is connected to the ground). The switch is added since the sudden surge of power to the peak detector causes it to have an overshot output in the beginning of a sampling period. To return the output to its normal level, a 'corrective discharge' is carried out by connecting the capacitor to the discharge resistor for a few milliseconds. Like the ADP194, the MAX323 is digitally controlled by the TelosB using a digital I/O pin.

The schematic of the expansion board (which contains the microphone, preamplifier, and peak detector) is shown in Figure 3.9.

Design parameters - sensing side

The first design parameter that we need to derive is the value of $\mathsf{C_{Ch}}$, or the size of the storage capacitor. The storage capacitor stores the voltage that corresponds to the maximum loudness of sound that has been observed within an active period. It is connected directly to the ADC port of the microcontroller, which samples the voltage level of the storage capacitor at the end of an active period. The storage capacitor is discharged after sampling so that it can start at a very low level in the next active period. It must be noted that the *storage capacitor* is different from the *boost capacitor*, the value of which will also be derived in a later subsection. The size of the storage capacitor depends on the charging speed required of the system, which in turn depends on the desired sound frequency range covered by the detector. The maximum rate at which the voltage across the storage capacitor can change is either

$$\frac{\mathsf{dV_{Ch}}}{\mathsf{dt}} = \mathsf{SR_1} \tag{3.5}$$

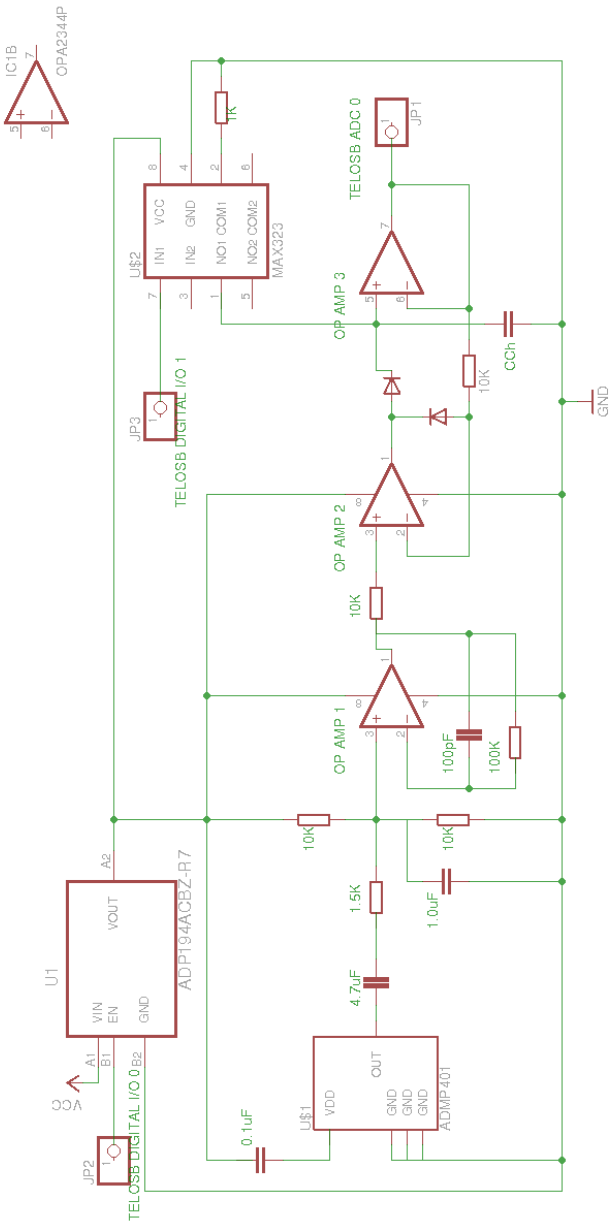


Figure 3.9: Design schematic.

or

$$\frac{\mathsf{dV_{Ch}}}{\mathsf{dt}} = \frac{\mathsf{IO_{max}}}{\mathsf{C_{Ch}}}, \tag{3.6}$$

whichever is smaller [38]. The values of $\mathsf{SR_1}$ and $\mathsf{IO_{max}}$, or the slew rate and short-circuit current of *OP-AMP 2* in Figure 3.9, are specified by the datasheet [152] as $\frac{0.8V}{\mu s}$ and 15 mA, respectively. The value of $\mathsf{C_{Ch}}$ is then dependent on the desired rate of voltage change. To derive this, we assume a maximum sound frequency of 10,000 Hz and require the system to be able to swing from the quiescent level to the upper rail (a 1.5 V swing) within a single cycle. A maximum frequency of 10,000 Hz is chosen for the initial design because most of the frequencies to which the human ear is sensitive can be found below 10,000 Hz [36]. Referring to Figure 3.10, assuming that $\mathsf{V_{Swing}} \approx \mathsf{V_{Amplitude}}$ and $\mathsf{t_1} \approx \mathsf{t_2}$, we can simplify the computation of $\frac{\mathsf{dV_{Ch}}}{\mathsf{dt}}$ as

$$\frac{\mathsf{dV_{Ch}}}{\mathsf{dt}} = \frac{\mathsf{V_{Swing}}}{2 \times \mathsf{t_1}} \tag{3.7}$$

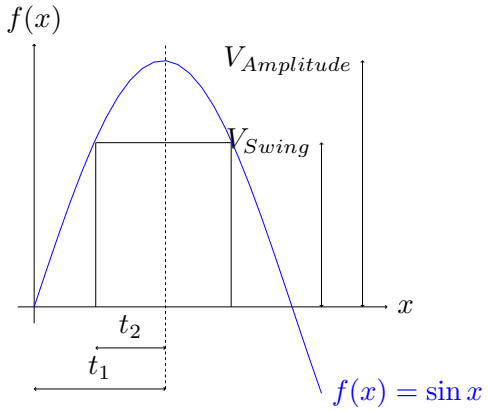


Figure 3.10: Diagram demonstrating simplifying assumptions adopted for parameter computation of $\mathsf{C_{Ch}}$ size.

Taking into account that $\mathsf{t_1}$ is $\frac{\frac{1}{10000}}{4}$, Equation 3.7 gives $\frac{30{,}000V}{s}$, which is smaller than $\mathsf{SR_1}$. This means that the speed is supportable by the operational amplifier subject to the correct $\mathsf{C_{Ch}}$ value. Substituting this value into Equation 3.6 and solving for $\mathsf{C_{Ch}}$ gives us 0.5 μF.

Ideally, the output of the peak detector should remain stable until a higher peak than the previous is seen in the input. In reality, the peak detector output decays over time due to leakage current (an effect called *voltage droop* [38]). The rate of discharge of the storage capacitor is

$$\text{Voltage droop} = \frac{\mathsf{I_{lk}}}{\mathsf{C_{Ch}}}, \tag{3.8}$$

where $\mathsf{I_{lk}}$ is the leakage current. There are five sources of leakage current, namely: capacitor leakage, printed circuit board leakage, op-amp leakage, diode leakage, and reset switch leakage.

With the exception of op-amp leakage and diode leakage, all components of $\mathsf{I_{lk}}$ are difficult to quantify. Diode leakage is negligible since we use an ultra-low-leakage diode in the design. That leaves the op-amp leakage or the input bias current of the op-amp, which the datasheet gives as 10 pA maximum. Thus, $\mathsf{I_{lk}} = 10$ pA. Using Equation 3.7 and the value derived for $\mathsf{C_{Ch}}$, this gives us a voltage droop of $\frac{0.00002V}{s}$.

The voltage droop simply gives the rate at which the output of the peak detector decays over time. The actual decrease in the output depends on how much time has elapsed since the peak value is stored by the circuit. Since we do not know the exact time at which the peak detector stores a new peak value, this introduces uncertainty in the output. This output is further affected by the length of the active period (which is dictated by the duty cycle, to be discussed later). Since the microcontroller does not sample the peak detector output until the end of an active period, the longer the active period, the more opportunity there is for the inaccuracy of the peak detector output value to increase - this is especially true for peak values that are recorded very early on in the active period.

Design parameters - power supply side

The equations defining the relationships between the charge time, the current draw length, the size of the boost capacitor and the duty cycle are presented and discussed in Chapter 2.4.

As already mentioned in Chapter 2.4, duty cycling is traditionally seen as a necessity imposed by the limited amount of harvested energy. As implied by Equation 2.5 however, the hardware itself imposes a cap on the duty cycle feasible. It is interesting to note that neither the storage size nor the solar panel output figures in Equation 2.5. Therefore, using larger batteries or larger solar panels will not help in solving the limit imposed by Equation 2.5.

If higher duty cycles than that imposed by Equation 2.5 are desired or needed, a possible solution will be to co-locate several sensor nodes. Enough nodes should be co-located so that the temporal coverage required is satisfied. If, for example, Equation 2.5 imposes a limit of 25% and 100% duty cycle is required, four nodes should be co-located, with the nodes taking turns in being active. The primary challenge in this solution will be the tight synchronization that will be required of the co-located nodes, and its main disadvantage will be the cost. A schematic for node co-location is shown in Figure 3.11.

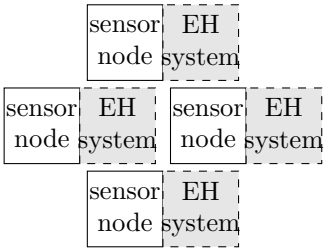


Figure 3.11: Node co-location schematic.

Another possible solution will be to use one node with several energy-harvesting systems, each with its own solar panel and energy storage device. Upon depletion, the node will simply switch energy-harvesting systems, allowing the previously used system to recharge. As with the previous example, if there is a 25% limit, but a 100% duty cycle is required, four energy-harvesting systems

will now be required. This will be less costly than the previous solution, and no synchronization will be needed, but there is the challenge of ensuring that the multiplexed power supply will be able to provide a supply level that is stable enough for continuous system operation. To the best of our knowledge, switched or multiplexed power supplies have never been used before in the context of overcoming duty cycle limitations in WSN nodes. The closest previous work is [118], where different subsystems in an embedded system are allowed to be powered simultaneously by different energy sources in an attempt to maximize energy harvesting efficiency. The schematic for node co-location is shown in Figure 3.12.

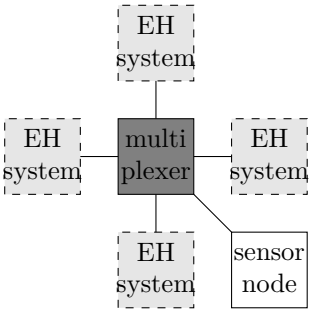


Figure 3.12: Power supply multiplexing schematic.

It must be noted that node co-location and power supply multiplexing only change the upper limit imposed by the physical system with regard to the maximum duty cycle achievable. The *actual* achieved duty cycle is still a function of the harvested energy and the energy usage pattern of the application. The amount of harvested (and stored) energy is affected by the node's location, harvesting efficiency, solar panel size, and storage capacity. The energy usage pattern of an application is usually tailored to the energy state of the system and the environment by power management algorithms. Power management algorithms are discussed next.

The microphone, preamplifier, and peak detector portion of our circuit are measured to have a collective current draw of 0.8 mA, translating to a duty cycle of 9.9%. We choose the boost capacitor size of 11,000 μF for a draw

time of 10.05 s and charge time of 91.0 s. We believe that such values give an acceptable trade-off between the length of the draw time and the interval between the draws.

3.5.2 Power management

For an actual deployment to function, the nodes need a power management algorithm. The power management algorithm manages the power consumption of the system, ensuring that system operation stays stable and functional despite variations in harvested energy. An obvious task for the power management algorithm will be adapting the system operation to the diurnal cycle, making sure that enough energy is harvested during the day so that the system remains functional even at night. The choice of power management algorithm will depend on several factors: the application type, the network topology, and the hardware available. We do not specify a specific algorithm since the node can be used for different types of applications and different network topologies. We do, however, discuss the factors that guide the choice and design of an appropriate power management algorithm.

Application type

Strictly speaking, a power management algorithm is any algorithm which changes an aspect of the system to meet energy constraints. What aspect is actually changed can vary from one algorithm to the next. For systems that regularly sample a single sensor, what is usually varied is the duty cycle. In [59], for example, the algorithm takes into account the predicted energy that will be harvested and assigns duty cycles to time periods called *frames*. The duty cycle assignment ensures that the node only consumes what it harvests, resulting in an infinite lifetime, or a mode of operation called *energy-neutral operation*. It also makes provision for when the actual harvested energy deviates from the predicted level, increasing the duty cycles in subsequent frames when more energy is harvested than predicted and reducing the duty cycles when less energy

is harvested. A power management algorithm such as that featured in [59] will be a possible good fit for systems that utilize our node, although modifications are necessary. For example, [59] assumes that there is a utility level associated with the duty cycle and that there is a maximum duty cycle beyond which the utility will no longer increase. Even without an explicit consideration of utility however, in our system, the hardware already explicitly imposes a limit to the duty cycle. Such a limit will have to be taken into account when using the algorithm in [59].

Other power management algorithms (such as [137]) assume that the processor or microcontroller is capable of dynamic voltage and frequency scaling (DVFS) and thus changes the speed of the processor or microcontroller in accordance with the energy state of the system. This is not applicable to our system since the microcontroller of the TelosB, like other low-power microcontrollers, does not have the DVFS feature.

A node can also possibly change the sensor or set of sensors being used in response to the energy state [3]. This is more applicable to nodes that use multiple sensors, which is currently not the case in our prototype (although it can be easily extended with other sensors; for instance, one can simply activate one or more of the onboard sensors of the TelosB).

Application or task versioning [128] is also a possible power management technique. In application versioning, the application running on the node changes depending on the energy state. This, however, may not be applicable to storage space-constrained systems like those used in WSNs.

Other algorithms facilitate power management by controlling the sequence or start time of tasks [112]. Algorithms that belong to this family are more suited for embedded systems that react to real-time events, rather than those designed for regular data gathering.

Network topology and configuration

The network topology and configuration also have an effect on the power management algorithm as it determines the additional tasks that a node has to do, in addition to sensing and sending its own data. The tasks change the characteristics of the application (for example, its periodicity) and the power management techniques that are applicable.

Topology-wise, networks can be divided into two: single-hop networks and multi-hop networks.

Nodes in multi-hop networks (except leaf nodes) have the additional task of acting as a relay for other nodes - therefore, they do not just send messages, but receive messages as well. How much energy is spent on this task depends largely on the number of other nodes the node is serving as a relay for, and the media access control (MAC) protocol utilized. The nodes can communicate either synchronously or asynchronously. In synchronous communication, message transmission consumes relatively little energy, but there is the challenge of regular clock synchronization (which consumes energy itself). Such an approach is utilized in [50]. Asynchronous communication like LPL does not need synchronization but can potentially consume more energy than synchronous communication (in LPL, the energy is usually spent by the sender, mainly in sending the prolonged preamble) and can possibly suffer from congestion.

Nodes in single-hop networks do not have to relay messages for other nodes, but depending on the terrain and deployment environment, such a topology may not always be feasible. Single-hop networks can be push-based, where nodes regularly send their data to a sink, or pull-based, where the nodes only send data after receiving a request to do so. The latter is usually used in systems where the base station is mobile and travels around the deployment area collecting accumulated data from the nodes.

Hardware

The hardware available for energy measurement or estimation affects the choice of power management algorithm. All power management algorithms assume the availability of some sort of information about the energy state of the system or the environment, and the availability and accuracy of that information depends on the hardware.

[59] for instance, assumes that the amount of energy being harvested is always known, a reasonable requirement given that the Heliomote [57], for which the algorithm in [59] is designed, is equipped with dedicated circuitry that measures the voltage and current (hence, power and energy) coming from the solar panels. Without dedicated hardware, nodes can measure the energy state of the system by measuring the voltage level of the energy buffer (usually a rechargeable battery or capacitor). It must be noted however that the battery voltage indicates the combined effect of the energy harvesting and the energy consumption. Unless all loads are turned off, the actual amount of energy being harvested cannot be known. Even if the loads can be turned off, the buffer voltage level usually rises very slowly in response to changes in energy level and can therefore only offer energy data of poor resolution.

3.5.3 Physical setup and testing

We create an expansion board for the TelosB incorporating the sensor and other circuit extensions (see Figure 3.13).

Basic system operation, as seen through the output of the peak detector (sensor) and the microphone, is plotted in Figure 3.14. In the graph, the sampling (or active) period starts at 57.8 s and ends at 69 s. The overshoot in the peak detector output at the beginning of the sampling period can be clearly seen, as well as its correction, brought about by the corrective discharge. Sound, produced by human speech, is generated at 60.0 s. The sound lasts for almost a second, and the peak detector output can be seen rising to the peak voltage

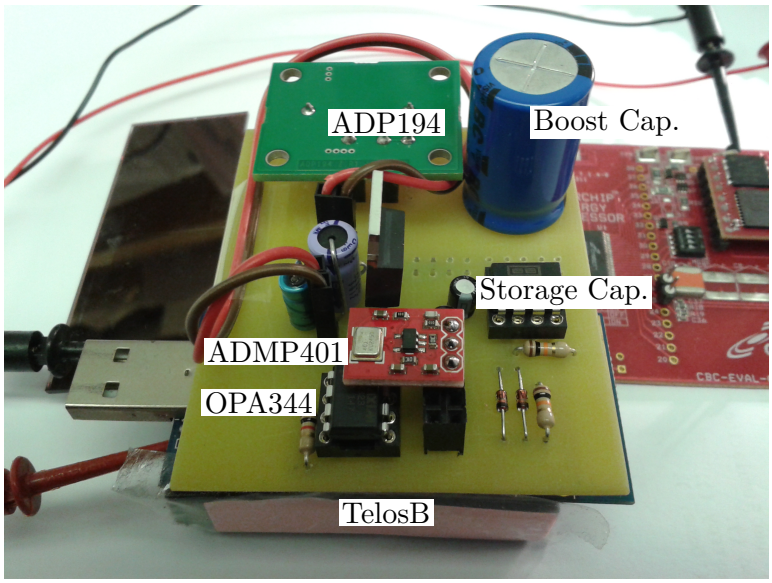


Figure 3.13: Expansion board, with the EnerChip and the solar panel in the background.

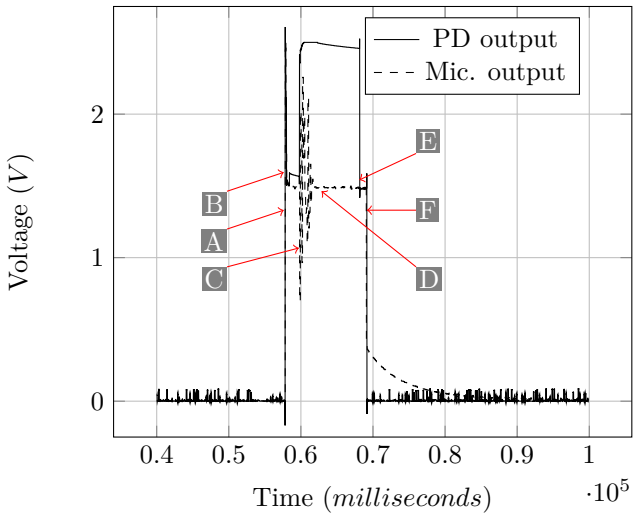


Figure 3.14: Sample operation. (A) Beginning of active period. (B) Corrective discharge. (C) Sound is introduced. (D) Cessation of sound. (E) Sampling and discharging of storage capacitor. (F) End of active period.

level generated by the sound. The retainment of the peak detector of the peak voltage can be seen, even after the cessation of sound generation. It must also be noted that the voltage droop exhibited by the system is larger than that computed in a previous section, suggesting underestimation of the leakage current,

or I_{lk}, in the computations.

The power consumption of the system (and the peak detector block alone), as it goes through the different phases of operation, is plotted in Figure 3.15. It can be seen that even when the peak detector block (which includes the MAX323 and the microphone) is not active, the system has a quiescent power consumption of 0.36 mW. When the peak detector block is active, power consumption rises, with a peak of 2.8 mW.

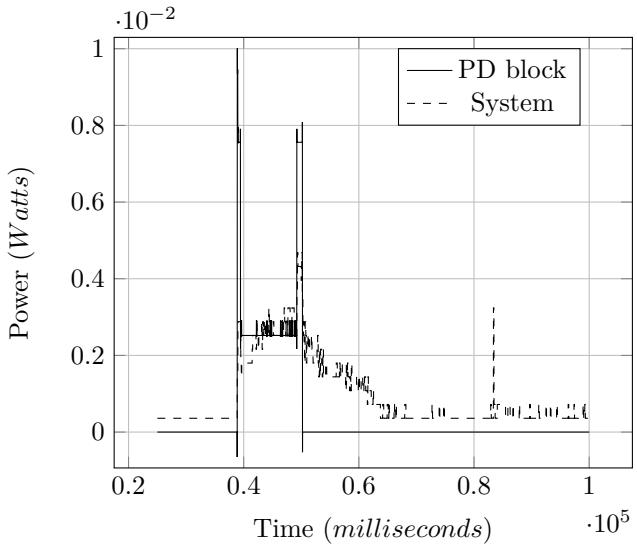


Figure 3.15: Power consumption.

To test the system, we expose the board to three different kinds of sound with different SPLs. We utilize three different kinds of sound: 10,000 Hz tone, white noise, and pink noise. White noise is a random signal with flat or constant power spectral density, and pink noise, on the other hand, has a power density which is inversely proportional to the frequency. We produce the sound in 1-s pulses fired in succession with inter-pulse gaps of random lengths. The possible inter-pulse gap lengths however, are limited in such a way that at least a single pulse will be fired within any active period of the system. This is so that 'soundless' active periods will not skew the average of the readings. Random gap lengths are used so that we can test the effects of differing sound introduction points in an active

period. The SPL is verified by a sound level meter (SLM) located very near to the node's microphone. In between each active period, a packet is sent to the base station: the packet contains the reading made in the preceding active period. The node is powered via energy harvesting in all of the experiments, with the solar panel illuminated by a lamp shining with a brightness of at least 1300 lx. Before any of the experiments, the solar panel and the energy harvesting system are exposed to the light for 50+ minutes without any load to ensure that the batteries are fully charged once the experiment begins. We run each experiment for a duration sufficient for the system to take 20 readings.

The values produced by the system in the experiments are plotted in Figures 3.16-3.18. It must be noted that Figure 3.16 has no data for 110 dB because of the limits imposed by our audio equipment. Figures 3.16-3.18 show that the system can differentiate between different SPLs. The graphs are sigmoid in shape, with the tapering in higher decibel values indicating the microphone's limitation in differentiating very loud sounds (those greater than 100 dB). Likewise, the baseline value and the first bars on the left-hand side show that the microphone can not detect sounds with SPL lower than 50-60 dB.

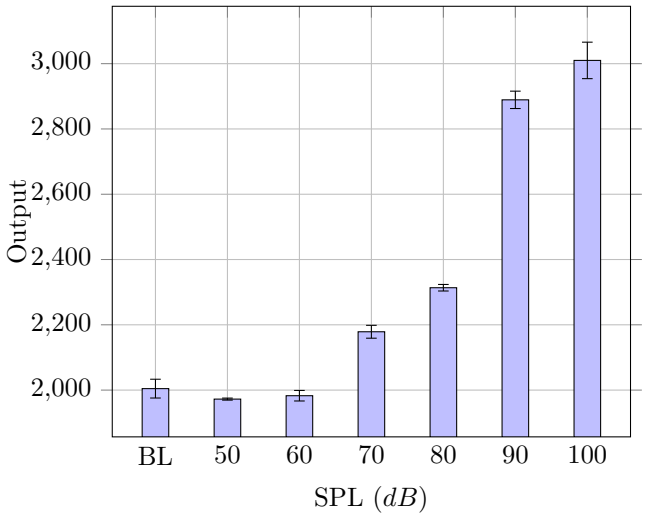


Figure 3.16: 10,000 Hz tone test. BL, baseline (no sound generated), all other numbers in the x-axis have dB as units.

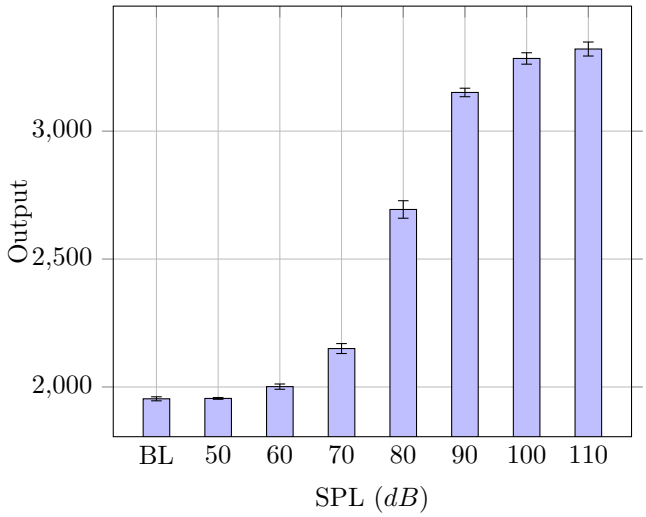


Figure 3.17: White noise test. BL, baseline (no sound generated), all other numbers in the x-axis have dB as units.

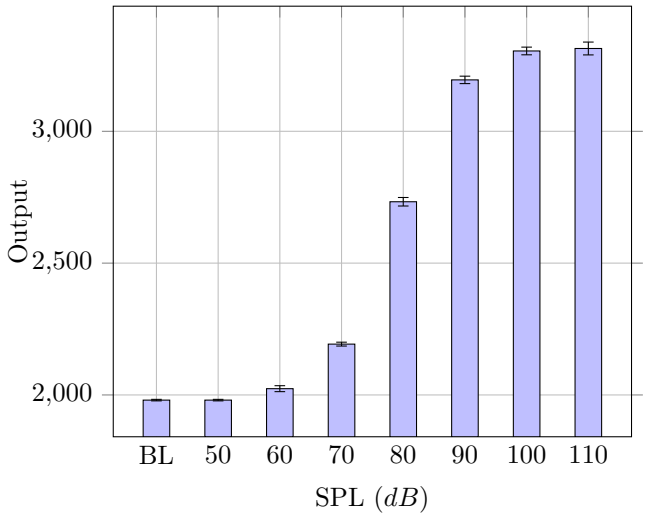


Figure 3.18: Pink noise test. BL, baseline (no sound generated), all other numbers in the x-axis have dB as units.

Using the data from the white noise experiment and exponential regression, we are able to find a function relating the sensor output to dB level. We remove the last data point (110 dB) since the resulting function no longer tracks the data points well. The resulting function is stated in Equation 3.9:

$$f(x) = 0.518203434173811 \times 1.00708470990609^{x} \tag{3.9}$$

The function, along with the experimentally-derived data points, is plotted in Figure 3.19. Below 70 dB, the function's maximum deviation from the actual value goes up to 7 dB. Beyond 70 dB, the maximum deviation goes down to 4 dB.

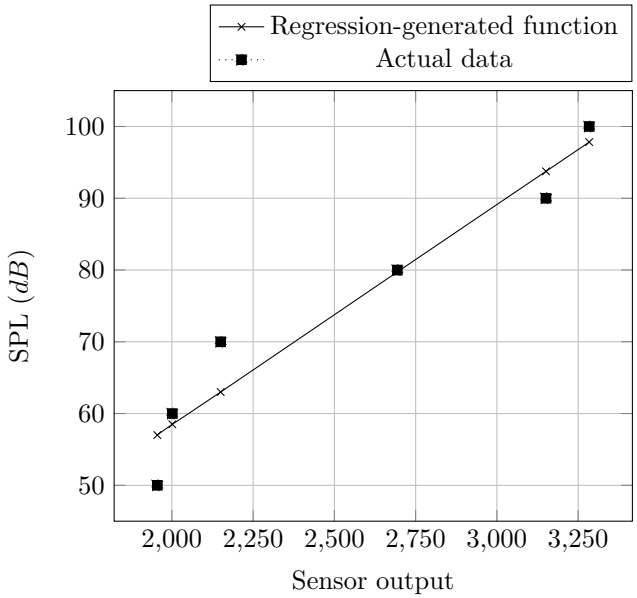


Figure 3.19: Function derived from curve-fitting the data points generated through experimentation. $f(x) = 0.518203434173811 \times 1.00708470990609^{x}$.

Theoretically, a table of finer resolution can be produced, which will enable a more accurate discernment of SPLs.

3.5.4 Related work

As discussed in Chapter 1, noise measurement using WSNs has been done in four previous studies: firstly, in the work presented in [130], [35], and [129]; secondly, in the work presented in [43] and [44]; thirdly, the work presented in [21] and [69], and most recently, our own work in Chapter 3.4.

Compared to the research presented in this subchapter, the first and second

sets of studies are more complete end-to-end solutions for noise monitoring: they employ A-weighting and have customized networking protocols (presented in [35, 44], respectively). The third study is notable for its use of fuzzy logic in inferring subjective noise annoyance from the sensor readings. The work done in Chapter 3.4 features the first noise-sensing WSN nodes to be powered by energy harvesting.

Table 3.2 summarizes the main differences between these previous studies.

Table 3.2: Comparison of previous studies

	Reported error range/accuracy	Notes	Platform	Year
[130], [35], and [129]	± 2 dB	first work on noise sensing; utilized dedicated hardware for noise sensing **and** metric computation	Tmote Sky/ TelosB [123]	2008
[43] and [44]	± 2 dB		CiNet [42]	2010
[21] and [69]	± 3 dB	used fuzzy logic-based method for inferring subjective noise annoyance	Sun SPOT [96]	2012
Chapter 3.4	—	energy harvesting (solar-powered)	TelosB [123]	2013
Chapter 3.5	± 7 dB	energy harvesting (solar-powered), based on peak detection	TelosB [123]	2014

In Chapter 3.4, we discuss the difficulties of noise sensing using low-power WSN nodes, difficulties that are compounded further by having the nodes powered by energy harvesting. These difficulties include the limited storage space available for the microphone voltage readings, and the limited word width of the system. Both place limits on the length of time over which $\mathsf{L}_{\mathsf{eqT}}$ can be taken. This work, in comparison, does away with $\mathsf{L}_{\mathsf{eqT}}$ and uses the peak level (as represented by the peak voltage amplitude generated by the sound) detected within a time period instead. This design decision, brought about in part through discussions with domain experts at New York's Center of Urban Science and Progress (CUSP), leads to the minimization of the need for high-frequency ADC sampling, making limited storage space and word widths much less of an issue. Therefore, the continuous and uninterrupted period of time over which the node can make its observation is much longer than that of the system in Chapter 3.4. While no noise codes currently employ peak levels as

their metric, it is still very useful in many situations and will complement other noise-sensing systems that employ $\mathsf{L}_{\mathsf{eqT}}$ as a measure. Moreover, our design being powered by energy harvesting makes it a good fit for low-maintenance urban sensing systems.

Just recently, Libelium released a Smart Cities sensor board [87] for the Waspmote [88]. The Smart Cities sensor board includes a linear displacement sensor for detecting cracks in building and infrastructures, a particle sensor for measuring air-suspended pollutants and a microphone for A-level weighted sound loudness. The Waspmote by default is battery-powered, although it does have provision for being powered by energy harvesting through solar panels. To the best of our knowledge, there are no known reports or studies detailing the performance of the Waspmote and the Smart Cities sensor board under energy harvesting conditions. A detailed comparison of the energy-harvesting Waspmote with the Smart Cities sensor board and an energy-harvesting TelosB with a custom sensor board is a possible future work.

Even without a detailed technical study however, one aspect of the competing systems can already be compared: cost. The Waspmote is a ready-for-deployment system. The cost of the Waspmote and Smart Cities sensor board combination (*sans* energy-harvesting component) is 270 Euros. In comparison, the TelosB is a prototype/research platform costing 77 Euros. The energy harvesting system that we use in our design is part of an evaluation board package, but a cursory check of individual component costs indicates that the total cost of components for the energy-harvesting system does not exceed 30 Euros. The cost of the customized sensor board comes in at under 20 Euros. All in all, the cost of a fully customized TelosB-based system (in a single board) is less than half the cost of an equivalent Waspmote-based system, even if only a modest number of units are produced.

3.5.5 Possible future work

Possible future work related to our design includes trying out new microphones to extend the range of sound intensities which the system can discern. It is also worthwhile to consider the incorporation of A-weighting to the design, either via a software implementation or a separate circuit.

Of primary concern in future design iterations is the voltage droop, which computations underestimated for the current design. Two strategies can be adopted: compensation and reduction.

We can opt to compensate for the voltage droop by adjusting the sampling strategy. In this work, a sample is taken at the end of the peak detector's active period. To compensate for the voltage droop, we can take several readings within an active period and just keep the highest among them. The main disadvantage that comes with this approach is the increased storage space utilization and operational complexity.

We can also aim for the outright reduction of voltage droop. This can be done by a better circuit board layout, or by using better op-amps. The storage capacitor size can also be increased (Equation 3.8), but in doing so, the range of frequencies detectable by the system will then decrease (Equation 3.6).

3.5.6 Summary

WSNs can potentially help in mitigating and preventing noise pollution which is increasingly becoming a concern in many cities all over the globe. Adding energy harvesting capability to WSNs can provide a low-maintenance, low-cost system to city administrators. Unfortunately, the metrics on which many currently existing noise codes are based are a poor fit to current energy-harvesting WSNs because of their limited capabilities. In this subchapter, we present a system which bases its output on peak levels, instead of $\mathsf{L}_{\mathsf{eqT}}$. While 'non-compliant' with existing noise codes, our design can potentially perform better than compliant energy-harvesting WSNs, at least when it comes to the length of

continuous observation time possible. It also provides a wider range of options which can be taken advantage of by power management algorithms, leading to better energy utilization. The significant potential of such a design (and the inspiration for it) came to us partly through discussions with city agencies in New York and London.

CHAPTER 4

Test methodologies: An indoor test methodology for solar-powered WSNs

4.1 Overview

Uniform test methodologies facilitate comparison and verification of results between researchers. This greatly aids in the advancement of the field, since researchers and designers are better able to build on top of each other's work. Unfortunately, no uniform test methodology exists for energy-harvesting WSNs, especially those that are actually implemented. In this chapter, we propose a test methodology for energy-harvesting WSNs that enable tests that are 'repeatable' (also something that most currently-used methodologies cannot do) and can be replicated by other researchers. Note that by 'repeatability' we mean the ability to test different algorithms, parameters, and hardware designs *several* times under *similar* energy-relevant conditions, leading to fair and verifiable comparisons.

4.2 Introduction and motivation

A significant challenge in power management algorithm and hardware design for solar-powered WSNs is the repeatability of experiments. Sunlight patterns are highly variable, seasonally varying in intensity and length, and strongly affected by local weather phenomena. Most power management algorithms for solar-powered WSNs are therefore tested via simulations (for example, [125] and [112]) or through simple tabletop experiments that involve lamps being turned on or off (for example, [63]). Both approaches overlook a great deal of detail that

is central for WSN operation. The accuracy and level of detail of the simulations are highly variable; they also frequently neglect hardware-specific features, such as non-idealities in the energy storage device characteristics. Most tabletop experiments (such as that documented in [63]) utilize actual hardware, but still omit the effects of daylight patterns: as a result, the nodes are possibly exposed to unrealistic conditions that they will never encounter in actual deployments (the ease with which this can be done will be demonstrated later in the chapter).

Repeatable experiments involving sunlight are also frequently needed in the field of photovoltaic (PV) cell (also called solar cell) design. In this domain, this problem is approached with the help of solar simulators (for example, [117]). Solar simulators generate light that approximates the intensity and the spectral content of natural sunlight. Most solar simulators however are designed for single-intensity exposure experiments: they subject the solar cells to a constant specific light intensity. They are not designed to automatically replicate the changing intensity of sunlight throughout the day. Moreover, solar simulators are expensive and bulky devices, with lighting elements that have very limited lifetimes.

In this work, we present a test methodology which enables repeatable indoor testing of solar-powered WSN nodes without the use of a solar simulator in the test itself. The test methodology induces the solar cell or solar panel to generate a level of power that it will exhibit under outdoor conditions at a specific time and place. The methodology has its basis on insights from PV cell design and astronomy. We firstly present the test methodology in an *apparatus-agnostic* manner, specifying the general principles of the test apparatus design. To prove its practicality however, we present our own *specific* design and implementation of the test apparatus. We present two variant designs for the apparatus. The first variant can be used in testing stand-alone wireless sensor network *nodes*. The second variant is a *distributed* version of the first, and can be used as a basis for testbeds designed to test wireless sensor *networks*. The performance and accuracy of the test methodology and our test apparatus are verified via

a series of experiments. Moreover, we also carry out a series of *demonstration experiments*, the likes of which will be useful in studies aiming for outdoor deployment of solar-powered WSNs. It must be noted that while this work focuses on WSNs and WSN nodes, the test methodology is also readily applicable to any solar-powered *embedded system*, even those that are not designed to be networked.

In summary, we make two primary contributions in this work. Firstly, the indoor test methodology, and the design for a *generic* test apparatus that can be used in its implementation; and secondly, the design of an actually-implemented apparatus. We present two variants of the apparatus: *centralized* and *distributed*. We also present the results of two significant sets of experiments: the first tests the performance of the test apparatus, while the second is comprised of experiments that demonstrate how the methodology can be used in determining hardware and software parameters for a WSN node.

The remainder of this chapter is structured as follows. The test methodology is presented in the following subchapter (Chapter 4.3). The design of a *generic* test apparatus on which the test methodology can be implemented is presented in Chapter 4.4. Chapter 4.5 discusses our *specific* design and implementation of the test apparatus. We test the performance of our test apparatus, and present the results of the said tests in Chapter 4.6. The results of two experiments that demonstrate the utility and possible applications of the test methodology are presented in Chapter 4.7; limitations are discussed in Chapter 4.8. Related work is discussed in Chapter 4.9. Chapter 4.10 presents possible future extensions of the research and summarizes the chapter.

4.3 Test methodology

Before outlining the general principles of the test methodology we need to understand the relationship between a surface's *irradiance*, the surface's location, and the current date and time; this will be discussed in Chapter 4.3.1. An un-

derstanding of a solar cell's 'state' and how such a state can be determined is also required and this will be discussed in Chapter 4.3.2. The general principles built from these two examinations are presented in Chapter 4.3.3.

4.3.1 Astronomical model of irradiance

Irradiance (unit $\frac{W}{m^2}$) is the total power from a radiant source falling on a unit area. A detailed discussion of how irradiance due to the sun is computed at a given location, time of day and time of year, is given in [108] and [70]. In this subchapter, we give a necessarily brief summary of the procedure.

The power density in sunlight which reaches the Earth from the sun is 1,367 $\frac{W}{m^2}$ [108]. However, some of this is absorbed in the atmosphere, so the power density that reaches the surface of the Earth is less. The amount of sunlight absorbed or scattered depends on the length of path through which sunlight has to travel to get to the surface. The path length is generally compared to a path directly vertical to sea level. This path length is designated as *1 atmosphere* or AM 1. At AM 1, after absorption is accounted for, the power density is reduced from 1,367 $\frac{W}{m^2}$ to 1,000 $\frac{W}{m^2}$. Assuming that the absorption constant depends only on the air mass, and taking I as the irradiance due to the sun we have Equation 4.1:

$$\mathsf{I} = 1,367(0.7)^{\mathsf{AM}} \tag{4.1}$$

However, [107] proposed that a better fit to observed data is obtained through Equation 4.2:

$$\mathsf{I} = 1,367(0.7)^{\mathsf{AM}^{0.678}} \tag{4.2}$$

$\mathsf{AM} = 1$ when the rays from the sun are directly over the surface in consideration. The AM for any ray source direction is given by Equation 4.3 (all angles stated in this subchapter are in *degrees*):

$$\mathsf{AM} = \frac{1}{cos\theta_z} \tag{4.3}$$

where θ_z is the zenith angle, which is the complement to 90° of the elevation α. The elevation is the angle between the sun's direction and the horizon.

$$\theta_z = 90 - \alpha \tag{4.4}$$

The elevation, or α, can be derived from Equation 4.5:

$$\alpha = sin^{-1}(sin\delta sin\theta + cos\delta cos\theta cos\omega) \tag{4.5}$$

θ is the latitude of the surface. δ is the solar declination, or the angle between the line joining the centres of the Earth and the sun, and the equatorial plane. This can be derived from Equation 4.6:

$$\delta = 23.45 \times sin(360 \times \frac{\mathsf{n} - 180}{365}) \tag{4.6}$$

where n is the day in the year ($\mathsf{n} = 1$ on 1^{st} January). ω is the hour angle or the angle difference between noon and the desired time of day in terms of a 360° rotation in 24 hours:

$$\omega = \frac{12 - \mathsf{T}}{24} \times 360 = 15(12 - \mathsf{T}) \tag{4.7}$$

where T is the time of day expressed with respect to solar midnight, on a 24-hour clock.

In summary, given the location of a surface, and the current date and time of day of interest, the irradiance due to the sun can be computed.

4.3.2 Solar cell state

Each solar cell has an associated set of current-voltage (IV) curves (Figure 4.1). The currently-relevant IV curve is dependent on the irradiance of the solar cell,

while the specific point on the curve (the solar cell's current *operating point*) is dependent on the electrical load imposed on the solar cell. The currently-relevant IV curve for the solar cell can be determined by measuring the cell's short circuit current (I_{SC}), which is unique for each level of irradiance. The relationship between the irradiance and the I_{SC} is highly linear [108]: for instance, if the I_{SC} at 1000 $\frac{W}{m^2}$ is 18.9 mA, the I_{SC} at 500 $\frac{W}{m^2}$ will be approximately 9.45 mA. We can consider the currently-relevant IV curve as the solar cell's *state*. It must be noted that how a solar cell is induced to reach a certain state is not relevant here. Two similar cells, exposed to two different light sources (different spectral content and intensities) can be said to be state-wise the same, as long as they have the same currently-relevant IV curve. This is the key concept which enables us to replace the lighting element in traditional solar simulators with a relatively low-power, less bulky, cheaper and more easily controlled alternative.

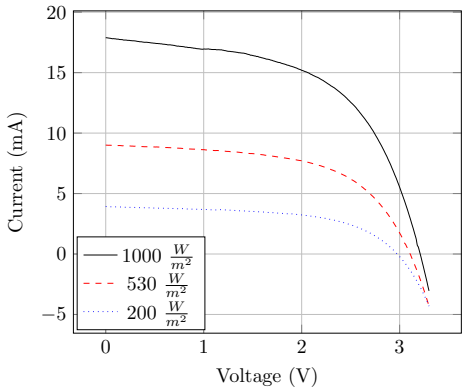


Figure 4.1: IV curves of a solar panel.

4.3.3 General principles

We are now in a position to state the general principles of the test methodology

1. Irradiance for a surface at a certain place, date, and time can be computed using the set of equations presented in Chapter 4.3.1. *Irradiance sequences* representing days (or parts of days) can then be generated by iteratively solving the set of equations.

It must be noted that in reality, the brightness of sunlight is continuously changing during the day - the irradiance therefore can actually be better modeled as a continuous function rather than a discrete sequence. However, we will be implementing the test methodology using discrete-time systems, so some discretization is necessary. We define the time interval between changes in irradiance values as the *temporal resolution* of a sequence.

2. The currently-relevant IV curve (hence, the irradiance of the solar cell) can be determined by measuring the solar cell's I_{SC}. By taking advantage of the linear relationship between the I_{SC} and the irradiance level, an irradiance sequence can therefore be converted to an I_{SC} *sequence*.

3. The simulation of a day (or part of it) can then be carried out by inducing the solar cell to sequentially produce the I_{SC} values specified in the I_{SC} sequence.

It must be noted that a single solar cell usually can not produce enough voltage or current to power an application. As such, what are used as energy harvesting devices are *solar panels*, which are similar solar cells placed in a series and/or parallel configuration to increase the output voltage and/or current. Like solar cells, solar panels also have IV curves, and their characteristics are derived from that of their component solar cells. Since most devices (including those used in this work) use solar panels and not simply solar cells, we refer to the energy harvesting device in subsequent subchapters as solar panels.

4.4 Generic test apparatus design

A generic test apparatus which can carry out the test methodology will need three components.

1. **Light source**. The light source's output does not need to have the same spectral content as sunlight. However, it must coincide (even if not com-

pletely) with the frequencies to which the solar panel is sensitive. The light source must be able to induce the solar panel to enter states corresponding to different irradiance levels, including the irradiance level of 1000 $\frac{W}{m^2}$, also known as *1 sun* (the maximum irradiance due to the sun possible at the surface of the Earth).

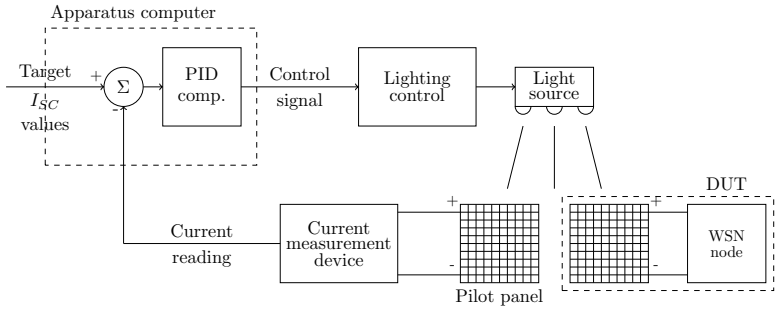
2. **Current measurement mechanism**. A generic test apparatus must be able to measure the solar panel's I_{SC}, or a suitable proxy to it. There are two challenges associated with measuring the I_{SC} - *where* it must be measured, and *how* it must be measured.

   The test methodology utilizes I_{SC} for determining the current state of the solar panel. However, measuring I_{SC} requires that the solar panel be in a short circuit, which is not the case when it is in a useful configuration (i.e., in a circuit). The problem can be solved by utilizing *proxy* I_{SC} measurements. Two measurements can serve as proxies for the I_{SC}: the I_{SC} of a pilot panel, and the output of the panel itself.

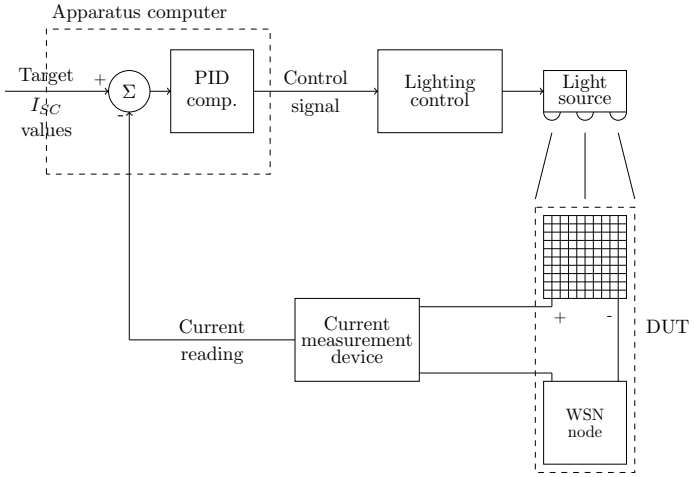
   We define a *pilot panel* as solar panel that is of the same type as the solar panel found in the device-under-test (DUT), and co-located with it. Assuming that the two panels have the same characteristics, and are under the same lighting conditions, they will be in the same state (since the currently-relevant IV-curve is defined only by the irradiance of the panel's surface). The I_{SC} of the pilot panel can then be measured and used in determining the state of both panels. The schematic of a generic test apparatus which uses the pilot panel I_{SC} as proxy for the DUT solar panel I_{SC} is shown in Figure 4.2a.

   The actual current of the solar panel can also be used as a proxy to a limited extent. This is due to the current value remaining constant (the same as the I_{SC}) for the most part of an IV curve (Figure 4.1, left-side portion). However, depending on the electrical load, this may not always be the case (see right-most part of Figure 4.1). The schematic of a generic

test apparatus which uses the actual solar panel current as proxy for the I_{SC} is shown in Figure 4.2b.



(a) Using the pilot panel I_{SC} as proxy



(b) Using the solar panel current as proxy

Figure 4.2: Generic test apparatus design schematic.

An additional challenge is the process of measuring the current itself. Accurate current measurement is a complex process. Most microcontrollers have analog-to-digital converters (ADCs) that measure voltage, but not current. This limits the prospect of the operation being done in distributed systems with low complexity. The current measurement process can be converted to a voltage measurement process with the use of shunt resistors. Shunt resistors however introduce losses, which may be unacceptable when the current is small to begin with. The losses due to a shunt resistor can be minimized by using smaller shunt resistor values. This is made fea-

sible using current sense amplifiers such as the LT6105 [90] or the TS1100 [106].

3. **Feedback mechanism**. A generic test apparatus must have a feedback mechanism which will adjust the lighting element's output in response to the difference between the actually-measured I_{SC} and the target I_{SC}. A feedback loop can be implemented using a Proportional-Integrative-Derivative (PID) controller [10]. The P, I, and D constants have to be carefully derived - improperly-set constants can lead to oscillations, with the output variable (the I_{SC} in this case) never converging to a specific value. The performance of the PID controller will also depend on the accuracy of the current measurements.

It should be noted that the design presented above is *generic*: it is a design which in theory, will enable *any* type of light source to be used in the apparatus (as long as the spectral content requirements stated above are satisfied). An actual implementation does *not* need to perfectly conform with the generic design - what is important is that the apparatus is able to induce the solar panel to produce the I_{SC} values contained in the I_{SC} sequence.

Our own implementation of the test apparatus features neither a current measurement mechanism nor a feedback loop. We justify the omission with our choice of a light source that is sufficiently *stable*. The details of our test apparatus design, our definition of stability, and its demonstration vis-à-vis our design, are presented next.

4.5 Test apparatus implementation

We construct two variants of the test apparatus: one for testing WSN *nodes*, which we call the *centralized* test apparatus (see Chapter 4.5.1), and another that can be used for testing *networks* of WSN nodes, which we call the *distributed* test apparatus (see Chapter 4.5.2).

4.5.1 Centralized test system

The lighting element in our implemented test apparatus (both centralized and distributed) is the Luxeon K 8-LED array (LXK8-PW40-0008 [91]). The light intensity of the Luxeon K is current-controlled by the buck regulator-controlled current source LM3406HV [154]. The output of the LM3406HV is then controlled via pulse width modulation (PWM). In the centralized version of our apparatus, the PWM signal is generated by an Arduino Uno [72] (interfaced to a PC via USB cable). In the distributed version, the PWM signal is generated by a TelosB WSN node [123]. For the centralized test apparatus, the Luxeon K is interfaced with a heatsink and mounted on a platform which enables the adjustment of the vertical distance between the LED array and the DUT. Figure 4.3 shows the schematic of our centralized test apparatus; the actual test apparatus is shown in Figure 4.4.

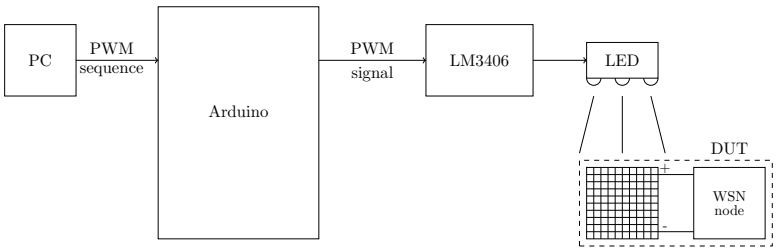
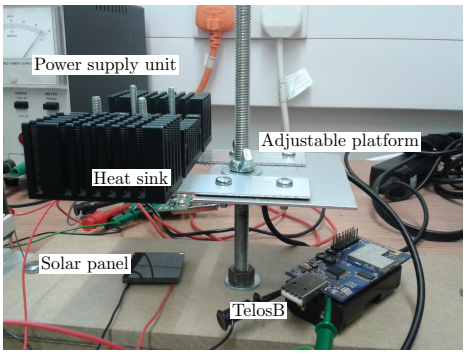
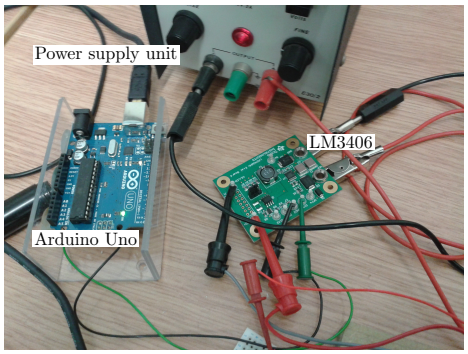


Figure 4.3: Design schematic of the implemented *centralized* test apparatus.

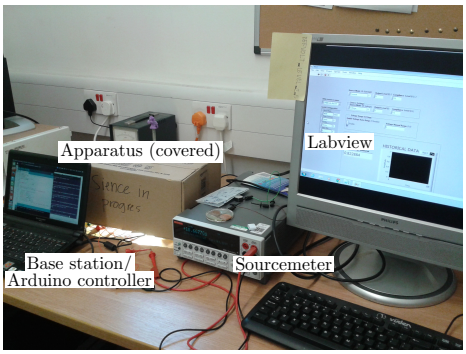
The feedback loop (and its current measurement input) is necessary for dynamically adjusting the lighting element in response to the difference between the measured I_{SC} and the target I_{SC}. This is particularly important for light sources with resulting I_{SC} values that tend to drift even with a constant light source setting. Some bulbs for example, glow brighter as their filaments heat up, even if the supply voltage remains constant. A feedback loop is not necessary in situations where the resulting I_{SC} for a given lighting element setting is distinct and unchanging. We call the tendency of a light source to have a distinct and unchanging I_{SC} at each setting its *stability*. We note that by 'light source' we mean the lighting element along with the mechanism which controls



(a) Lighting element (under the heatsink) and vertically-adjustable stand



(b) Arduino and LM3406



(c) Test apparatus in operation, with base station and sourcemeter

Figure 4.4: Centralized test apparatus implementation.

its settings. In our specific setup, the light source is collectively comprised of the LED array, the LM3406HV, and the Arduino.

We can quantitatively define stability using two metrics. Firstly, we can measure the *variance* of the solar panel I_{SC} observed at a given setting. The variance provides a measure of how much the I_{SC} fluctuates. The lower the variance, the more stable the light source is at that setting. A *perfectly* stable light source has a variance of 0 for all sets of I_{SC} readings. Secondly, we can measure the tendency of the I_{SC} to rise or fall over time using the *trendline slope*. Using linear regression, we generate a trendline for the values observed at a given setting, and take note of its slope. This is done for all settings tested. The closer to 0 the trendline slope is, the more stable the light source is at that setting. A *perfectly* stable light source has a trendline slope of 0 for all sets of

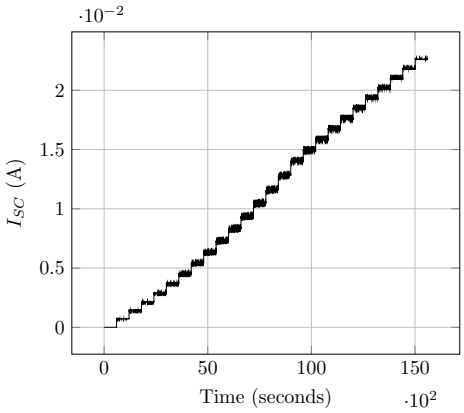
I_{SC} readings.

The Arduino allows the user to specify the duty cycle of the PWM signal with 256 values (0 = 0% duty cycle, and 255 = 100% duty cycle). This corresponds to 256 lighting element settings. To test the stability of the light source (vis-à-vis our recent definition) we test 26 PWM values (multiples of 10, 0 to 250), and measure the resulting solar panel I_{SC}. The vertical distance between the solar panel and the LED array is set to 60 mm, and the apparatus is covered during the tests to control the lighting. The test is run continuously, with the PWM value increased in a monotonic manner. The I_{SC} is measured using a Keithley 2401 [86] sourcemeter connected to a PC running NI Labview [93]. The Labview program collects a current reading every second, resulting in 600 readings per PWM value tested. The measured I_{SC} values are plotted in Figure 4.5a. Figure 4.5b shows the plot of the measured I_{SC} against the PWM values. Figure 4.5b is generated by taking the mean of the 600 current readings taken for each PWM value.

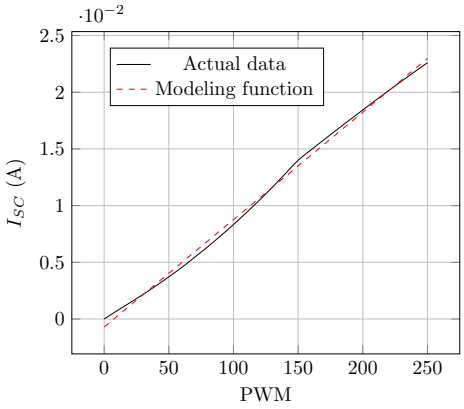
The variance and trendline slope for each set of I_{SC} readings collected are plotted against PWM values in Figure 4.6a and Figure 4.6b, respectively. It can be seen in Figure 4.6 that the light source is *not* perfectly stable. Nevertheless the light source is still *highly* stable, exhibiting a maximum variance of 1.825×10^{-9} (at PWM value 140), and a maximum trendline slope of $2.3 \times 10^{-7} \frac{V}{s}$ (at PWM value 140).

The extent to which a light source's stability is acceptable is affected by the temporal resolution of irradiance sequence, or the time interval for which a certain irradiance value holds. A larger interval (poorer temporal resolution) gives more opportunities for the I_{SC} to average out and thus tolerate more instability. The experiments are carried out utilizing 5-minute intervals (a 24-hour period is divided into 288 sub-periods). This is long enough, we believe, to accommodate the minimal instability of the light source.

We take this opportunity to underscore a point made earlier in the introduction about the inadequacy of simple tabletop experiments that do not consult astronomical models. The solar panel we utilize is measured to have a 1 sun I_{SC}



(a) Plot of I_{SC} as PWM is monotonically increased



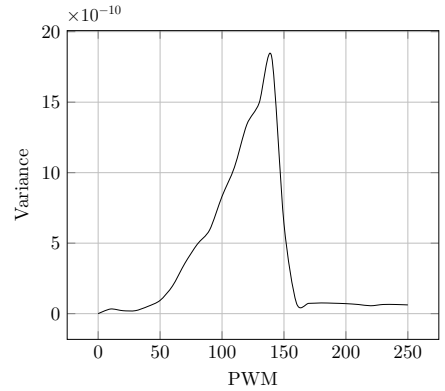
(b) Plot of I_{SC} vs PWM, both from averaged empirical data and a linear regression-produced function

Figure 4.5: Plots for characterizing the solar panel - *centralized* test apparatus configuration.

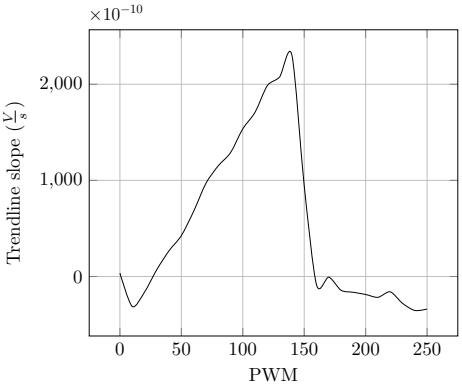
of 18.9 mA (some of its IV curves are plotted in Figure 4.1). The measurement is done using a calibrated Newport Oriel Sol3A Class AAA 4 x 4 inch solar simulator [117]. As can be seen in Figure 4.5b however, beyond the PWM value of 210, the I_{SC} already exceeds 18.9 mA. This shows how easy it is to place the solar panel in a state it will never be in under natural outdoor conditions.

If the light source is stable, 'day simulation' will simply consist of sequentially placing the light source at the appropriate settings. To do this the I_{SC} sequence will have to be converted to a *settings sequence* which will then be 'played'. For our apparatus, the settings sequence is called the *PWM sequence*.

To convert the I_{SC} sequence to a PWM sequence, we first need to find a



(a) Variance vs PWM



(b) Trendline slope vs PWM

Figure 4.6: Stability metrics - *centralized* test apparatus.

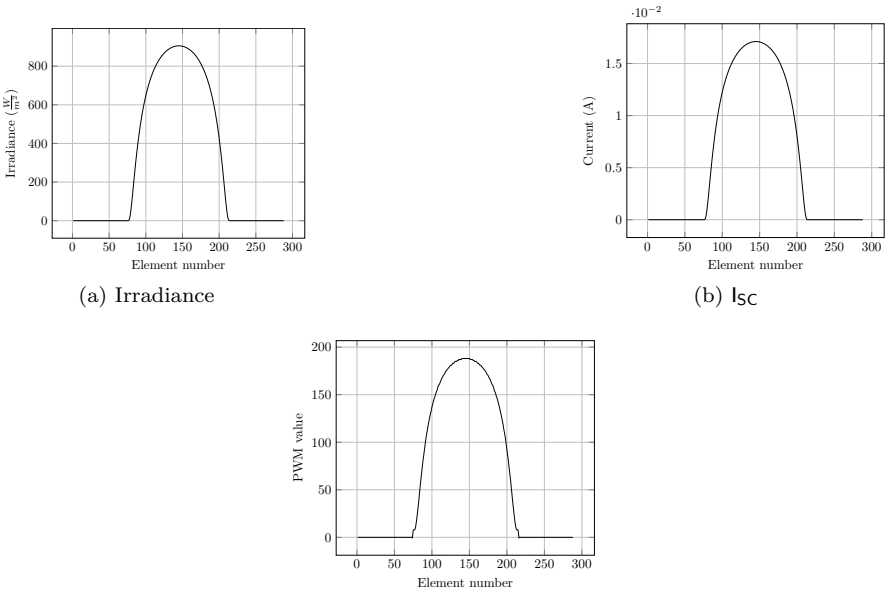
function that will approximate the empirical data in Figure 4.5b. This can be done in several ways. It is desirable that the function complexity be minimized as much as possible without sacrificing too much accuracy. To this end, we adopt a function generated through linear regression (Equation 4.8):

$$f(x) = 0.000094738695 \times x - 0.000707917246 \tag{4.8}$$

Equation 4.8 is also plotted in Figure 4.5b. We can then take the *inverse* of Equation 4.8, resulting in a current-to-PWM function. By rounding the output of the current-to-PWM function, an $\mathsf{I_{SC}}$ sequence can be converted to a PWM sequence.

As an example of the transformation between the three types of sequences,

Figure 4.7 shows the irradiance sequence, I_{SC} sequence, and PWM sequence for September 21 2014 in London, England (coordinates 51.5072° N, 0.1275° W). The astronomical model is evaluated with 5-minute intervals, resulting in a 288-element sequence for the 24-hour period.



(a) Irradiance

(b) I_{SC}

(c) PWM

Figure 4.7: Three sequences for September 21 2014 (autumnal equinox), London.

4.5.2 Distributed test system

There are some tests for which a single test apparatus will not suffice. For example, a single test apparatus cannot test the effect of differing sunlight patterns within a deployment area on network performance. Such tests are important as deployed nodes will likely receive different sunlight patterns because of several factors, including for example foliage, shadows or cloud cover. To carry out such tests (which are tests on *networks*, and not simply network *nodes*), we will need several instances of the test apparatus working together.

Arguably, the apparatus already presented can also be 'forced' to test networks. Several instances of the apparatus can be connected using USB cables,

and while USB cables are severely limited in length, the power level of the WSN node radios can frequently be adjusted, enabling the implementation of multi-hop networks even within a single room. Nevertheless, this is an unsatisfactory solution as it misses many aspects of actual deployments: for instance, the effect of the environment on topology through radio wave diffraction, diffusion and refraction.

To this end, we create a version of the test apparatus specifically for testing networks of nodes. Instances of the test apparatus are controlled by a base station, effectively forming a distributed system. Using the original settings sequence as basis, a new sequence can be customized for each test apparatus. To simulate the effect of permanent occlusion such as being under foliage, all the values in the PWM sequence can be attenuated by a certain factor. Temporary occlusion such as temporary cloud cover can be simulated by limiting the attenuation to certain periods. An instance of the test apparatus can be located anywhere within a building, as long as it is close to an electrical socket and within communication range of at least one other test apparatus which will enable it to form a multi-hop path to the base station. Our system can be used in constructing an indoor WSN *testbed* similar to Motelab [160] and Indriya [27], but tailor-made for solar-powered WSNs.

The primary differences between the test apparatus designed for node testing and the test apparatus designed for network testing are in the distribution of the PWM sequence and the generation of the PWM signal. In the test apparatus designed for node testing, PWM signals are generated by an Arduino, which receives the PWM sequence from a PC via a USB cable. In the test apparatus designed for network testing, the PWM signals are generated by a TelosB WSN node, while the PWM sequences are wirelessly received by the same WSN node from the base station. Our base station consists of a PC connected via USB to a TelosB WSN node.

Another difference between the two designs is in the power supply. The centralized version uses a variable-voltage bench power supply, while the dis-

tributed version uses a dedicated fixed-voltage power supply unit that is easier to transport. The TelosB WSN node of the distributed test apparatus is currently powered by batteries, although in future design iterations changes will be made so that it can also be powered from the fixed-voltage power supply unit.

While the test apparatus designed for node testing uses a box cover for keeping out ambient lighting, the distributed test apparatus uses a dedicated test chamber.

For communicating PWM sequences and synchronization information, the WSN nodes (both at the base station and those at the test apparatus instances) use the Tymo routing protocol/service. Tymo [105] is the TinyOS [103] [47] implementation of the Dynamic MANET On-demand (DYMO) routing protocol [53]. Dymo is a routing protocol designed by the Internet Engineering Task Force (IETF) to enable dynamic point-to-point routing between mobile nodes. It was originally designed to run on top of the Internet Protocol (IP).

The schematic of the distributed test apparatus is shown in Figure 4.8. Figure 4.9 shows the actual apparatus.

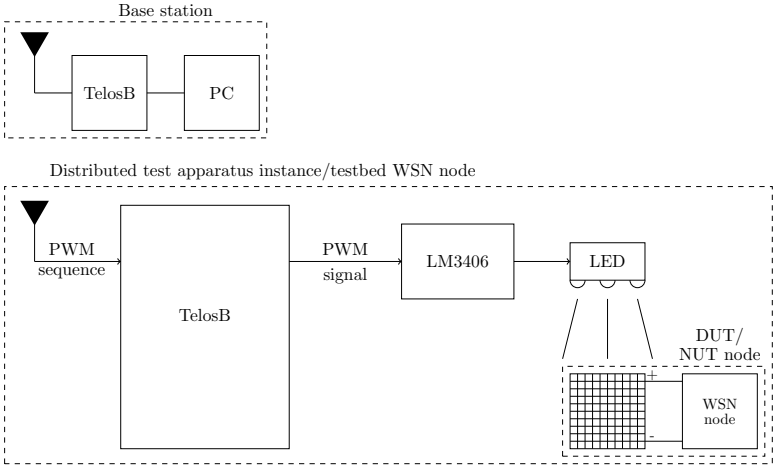


Figure 4.8: Design schematic of the implemented *distributed* test apparatus.

Note that when using the distributed test apparatus, we effectively end up with *two co-located WSNs*: one for the test bed/distributed test apparatus (composed of TelosBs in our implementation) and the other the network-under-test

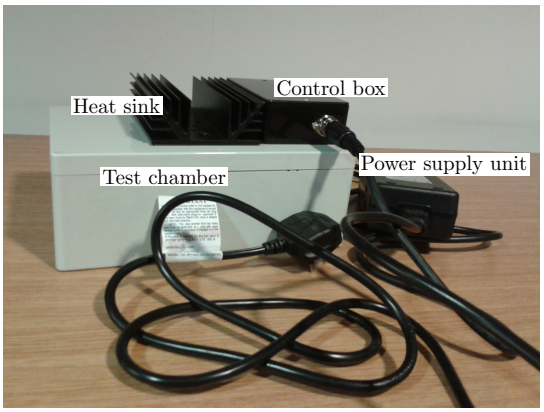


Figure 4.9: Distributed test apparatus implementation. TelosB and LM3406 inside control box (beside the heatsink).

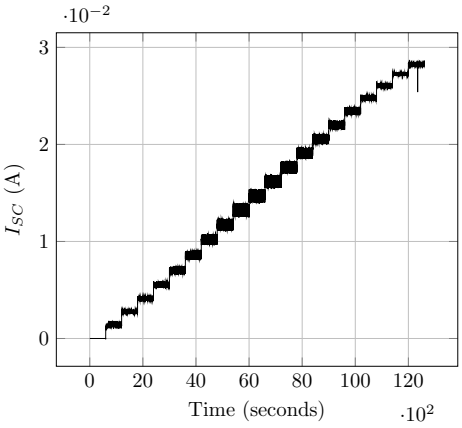
(NUT). The members of the NUT (labelled 'WSN node' Figure 4.8) need *not* be TelosBs. The co-location of two WSNs opens up the possibility of interference, which must be minimized. To achieve this, we employ two measures.

Firstly, the communication between the nodes of the testbed WSN is sparse. Such communication only happens at the beginning of the simulation when the patterns are being distributed, and at the end of the simulation, when the nodes report back to the base station. The post-simulation report enables the base station to ascertain that the simulation was successfully carried out by all nodes. We do not deal with the issue of clock drift or clock synchronization in our current implementation.

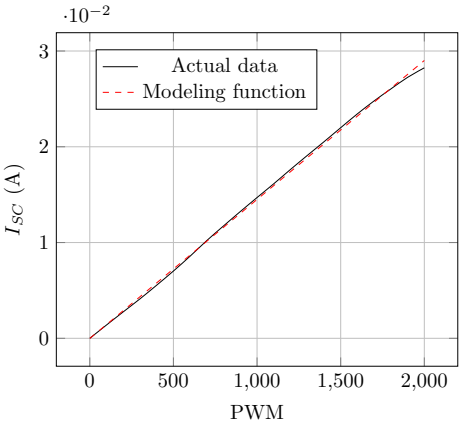
Secondly, we recommend that the two WSNs use different radio channels or frequency sub-bands. The ability to choose channels is a readily-available feature even in low-power radios such as the CC2420 [155], the radio module used in the TelosB. Additional isolation measures were not felt necessary.

To enable the conversion of $\mathsf{I_{SC}}$ sequences to PWM sequences suited to the distributed test apparatus, we repeat the experiment performed on the centralized test apparatus. The resulting $\mathsf{I_{SC}}$ as the PWM duty cycle is increased is shown in Figure 4.10a. It should be noted that Figure 4.10a has differences from Figure 4.5a. This is to be expected, since the vertical distance between the solar panel and the LED array is different in the two apparatuses (50 mm

vs 60 mm). Another difference between the two plots is the range of PWM values. We implement the PWM function of the TelosB using TinyOS' Alarm mechanism, setting the period of the PWM signal to be 2.04 ms, or the same as that of the Arduino PWM signal. However, unlike in the Arduino where the PWM duty cycle can be specified in 256 levels, in our implementation, one can specify the PWM duty cycle in 2041 levels (0 to 2040, where 0 = 0% duty cycle, and 2040 = 100% duty cycle). We test the PWM values by increments of 100, from 0 to 2000. The results of averaging the I_{SC} values and plotting the averages against the PWM values are shown in Figure 4.10b.



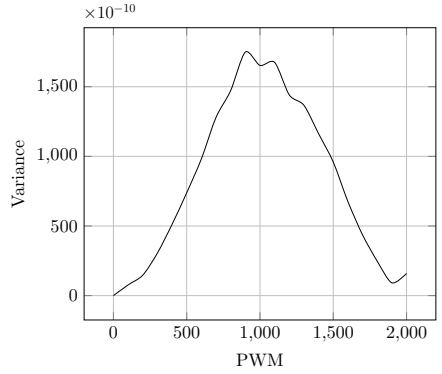
(a) Plot of I_{SC} as PWM is monotonically increased



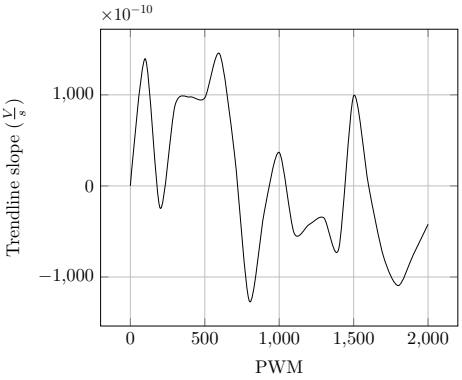
(b) Plot of I_{SC} vs PWM, both from averaged empirical data and linear regression-produced function

Figure 4.10: Plots for characterizing the solar panel - *distributed* test apparatus configuration.

We also plot the variance and trendline slope for each set of I_{SC} readings collected in Figure 4.11a and Figure 4.11b, respectively. The light source of the distributed test apparatus exhibits a maximum variance of 1.74885×10^{-7} (at PWM value 900), and a maximum trendline slope of $1.44899 \times 10^{-7} \frac{V}{s}$ (at PWM value 600). Similar to the centralized apparatus, the light source of the distributed apparatus is *not* perfectly stable, but it is sufficiently stable, we believe, for our application.



(a) Variance vs PWM



(b) Trendline slope vs PWM

Figure 4.11: Stability metrics - *distributed* test apparatus.

We model the empirical data in Figure 4.10b using a function generated through linear regression (Equation 4.9), also plotted in Figure 4.10b. The inverse of Equation 4.9 can be used in generating PWM sequences suited for the distributed test apparatus.

$$f(x) = 0.000014511145 \times x - 0.000013327296 \tag{4.9}$$

The operation of the distributed test apparatus can be divided into four phases, diagrammed in Figure 4.12:

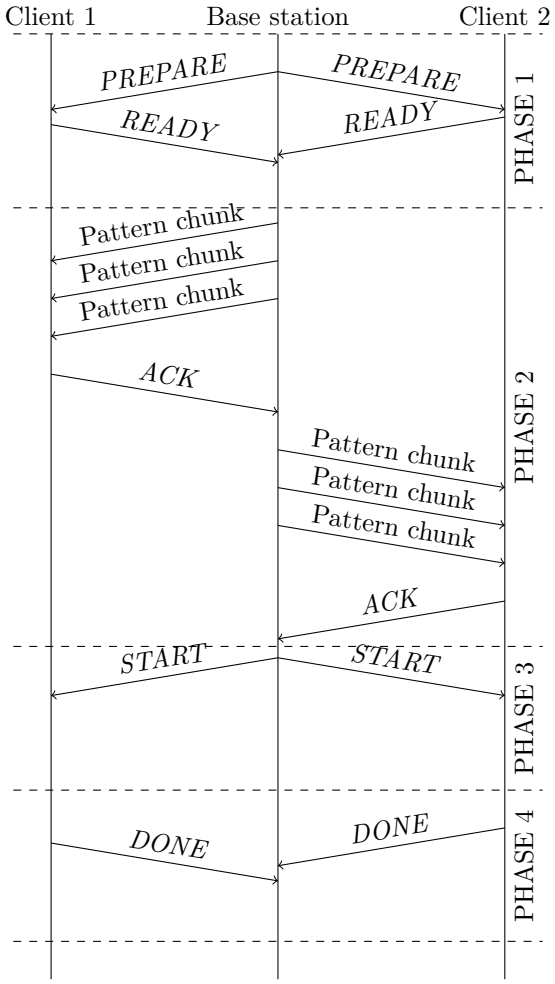


Figure 4.12: Distributed apparatus operation diagram.

1. **PHASE 1.** In the first phase the base station broadcasts a *PREPARE* message to the clients. The *PREPARE* message also contains the number of packets that will comprise a complete PWM sequence. A client receiving a *PREPARE* message will then respond with a *READY* message,

signifying that it is ready for Phase 2. Figure 4.12 shows a setup with a single base station and two clients. While Figure 4.12 shows client 1 and client 2 being within direct communication range of the base station, the operation discussed here also applies to clients that are not within direct communication range of the base station. A *PREPARE* message is routed by other nodes to clients not within direct communication range of the base station. *READY* messages are routed from the said clients to the base station in the same way. The base station takes note of the *READY* messages that it receives, making sure that all clients are accounted for. If nothing is heard from one or more client nodes, the *PREPARE* message is broadcasted again one more time. If the set of clients that responded is still incomplete, the user is informed and system operation is halted. Once a complete set of *READY* messages is collected, system operation proceeds to the second phase.

2. **PHASE 2.** In the second phase, the base station sends to each client the PWM sequence it would need to 'play' later. The sequence is sent over a span of several packets (differentiated through sequence numbers), the exact number of which depends on the length and resolution of the test desired by the user. The sequence *can* be unique for each client. The client will know that it has received a complete sequence by counting the number of pattern-comprising packets it has received and comparing the said number with the number it received earlier through the *PREPARE* message. After receiving a complete sequence, the client sends an acknowledgement (*ACK*) packet to the base station, signifying that it could move on to the next client. If no acknowledgement is received by the base station, it will resend the entire sequence. Once all sequences are sent (and acknowledged), system operation proceeds to the next phase.

3. **PHASE 3.** In the third phase, the base station broadcasts a *START* message to all clients. This causes each client to start 'playing' the sequence

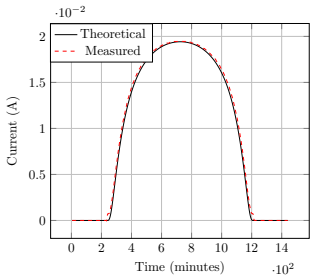
it received in the second phase. The system does not utilize any time synchronization mechanism, so the clients could potentially start playing out the patterns at slightly different times. Nevertheless, the differences will be less than a second at most, and thus will affect the accuracy of the test only very slightly.

4. **PHASE 4.** In the fourth phase, the clients send *DONE* messages to the base station after playing the sequences assigned to them. The *DONE* messages are used by the base station (and the user) in ensuring that all client nodes finished playing the patterns assigned to them, and that the test is successfully finished in its entirety.

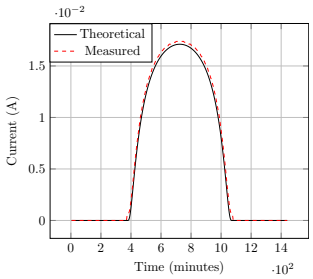
4.6 Test apparatus performance

The primary purpose of a test apparatus is to induce the solar panel to sequentially produce I_{SC} that correspond to the irradiance values determined by the astronomical model. To test the effectiveness of our test apparatus implementations, we run simulations corresponding to 3 different dates and compare the resulting I_{SC} values to those in the model-generated sequence. The 3 days simulated are June 21 2014, September 21 2014, and December 21 2014. These dates correspond to the northern hemisphere summer solstice, autumnal equinox, and winter solstice, respectively. The location assumed is the city of London. The astronomical model is evaluated with 5-minute intervals, with the sequence beginning at midnight of the specified day and ending just before the beginning of the next day. The same solar panel (1 sun I_{SC}: 18.9 mA) and apparatus setup (DUT-LED array distance: 60 mm for centralized, 50 mm for distributed) as those used in Chapter 4.5 are used. Consequently, the I_{SC}-PWM functions presented in Chapter 4.5 are used in generating the PWM sequences. A Keithley 2401 sourcemeter controlled by Labview measures the short circuit current at 1-second intervals. The readings are averaged over 5-minute sets (300 readings in each set). The averaged readings from the centralized test apparatus, along

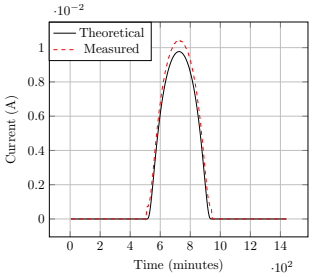
with the target I_{SC} values, are shown in Figure 4.13. We plot the difference between the theoretical and measured values in Figure 4.14a.



(a) Summer solstice



(b) Autumnal equinox

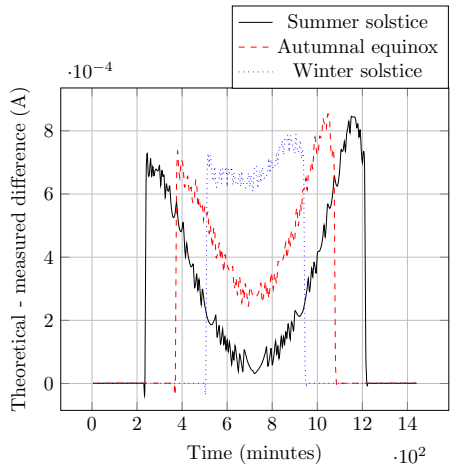


(c) Winter solstice

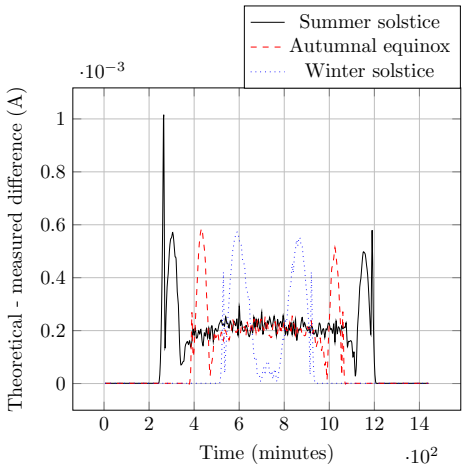
Figure 4.13: Measured and theoretical (target) I_{SC} values, *centralized* test apparatus.

In Figure 4.13 and Figure 4.14a it can be seen that the measured I_{SC} values follow the theoretical I_{SC} values very closely. Between the five minute averages, the *single* maximum difference between measured value and theoretical value is 0.84 mA for the summer solstice (with a mean of 0.245 mA, 0.36 mA if night portion is excluded), 0.85 mA for the autumnal equinox (with a mean of 0.242 mA, 0.49 mA if night portion is excluded), and 0.789 mA for the winter solstice (with a mean of 0.2 mA, 0.423 mA if night portion is excluded). For all setups, the maximum deviation occurs in the lowest non-zero PWM values - at sunrise or sunset. Referring to Figure 4.5b (left-side portion), this is to be expected, as this corresponds to the point where the modelling function deviates the most from the empirical data. Another possible source of the deviation is the imperfect stability of the light source.

A prominent feature of Figure 4.13c is the deviation between theoretical and



(a) Centralized apparatus



(b) Distributed apparatus

Figure 4.14: Difference between theoretical and measured currents.

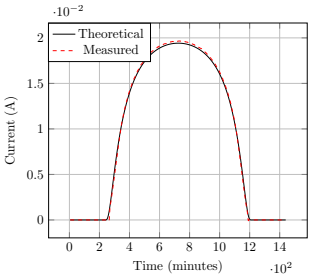
measured I_{SC} in the mid-day section. This can also be explained by the deviation between the empirical data and the modelling function. A closer inspection of Figure 4.13a and Figure 4.13b reveals deviation of roughly the same magnitude in the same I_{SC} range (near 10 mA). However, the sequences for the summer solstice and the autumnal equinox spend significantly less time at the said I_{SC} range compared to the winter solstice sequence, so the effect is significantly less overall. The precision therefore with which the test apparatus can replicate an I_{SC} sequence closely depends on the values contained in the sequence. For our specific setup, an I_{SC} sequence with more values that translate to very low non-

zero PWM values or are close to 10 mA will tend to be replicated less precisely than those with less.

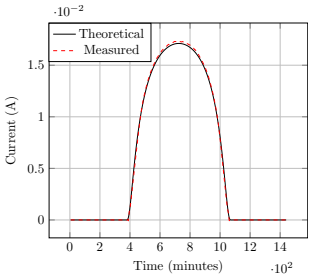
Before we discuss how the distributed version performs in replicating the I_{SC} sequences, we note that the distributed apparatus cannot properly generate light for PWM values lower than 50. Instead of dimmed light, what the LED produces at PWM values 0 - 50 is light with maximum level of brightness interspersed with very brief periods of darkness. This is not apparent in Figure 4.10a, since the PWM values in Figure 4.10a are tested with increments of 100 beginning at 0. We attribute the behaviour to how TelosB generates the PWM signal. While the Arduino runs its program 'bare metal', TelosB runs TinyOS, which is a simple multi-tasking operating system. The PWM signal is produced by the repeated execution of two tasks. The first task coincides with the beginning of a PWM period, and is in charge of pulling the output pin to a value of **HIGH**. The second task is executed later, and pulls the value of the output pin to **LOW**. The interval between the two tasks depends on the duty cycle or PWM value. We use the Alarm mechanism to facilitate the timing of the two tasks. However, at very low PWM values, the interval is too small that the second task is not consistently executed. This leads to periods where the pin value is never pulled down, or where the duty cycle generated is effectively 100% (hence, maximum brightness). This inconsistency is caused by the limitations of the TinyOS Alarm, scheduler, or both. To address this limitation (which would lead to significant errors for the apparatus), we round down all values in the PWM sequence that are lower than 50 to 0.

The measured I_{SC} values for the distributed test apparatus are shown in Figure 4.15. We also plot the difference between the theoretical and measured values in Figure 4.14b. The distributed test apparatus is accurate in replicating the sequences for all dates tested. Between the five minute averages, the *single* maximum difference between measured value and theoretical value is 1.01 mA for the summer solstice (with a mean of 0.155 mA, 0.229 mA if night portion is excluded), 0.585 mA for the autumnal equinox (with a mean of 0.1 mA, 0.218

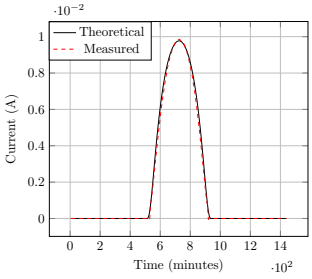
mA if night portion is excluded), and 0.571 mA for the winter solstice (with a mean of 0.077 mA, 0.156 mA if night portion is excluded). In general, the distributed version outperforms the centralized version when it comes to accuracy. This can be attributed to several factors. The distributed apparatus' lighting mechanism is superior to that of the centralized apparatus in terms of the trendline slope (although it exhibits greater variance). The distributed apparatus' modelling function is also a slightly better fit to the data it models compared to its counterpart in the centralized apparatus. Finally, PWM values in the distributed test apparatus are represented with the range 0-2041, compared to 0-255 in the centralized apparatus. A larger range of numbers translates to greater precision in specifying the PWM signal's duty cycle.



(a) Summer solstice



(b) Autumnal equinox



(c) Winter solstice

Figure 4.15: Measured and theoretical (target) I_{SC} values, *distributed* test apparatus.

4.7 Demonstration experiments

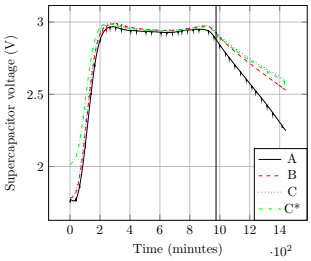
To demonstrate the full benefits of the test methodology and the test apparatus, we conduct two experiments the likes of which are important prior to deploying solar-powered WSNs. We use the centralized version of the test apparatus in the experiments. The energy harvesting component of our setup is a 30 mm by 30 mm solar panel with I_{SC} of 18.9 mA (the same as that used in Chapter 4.5). The solar panel is connected to a 50 F supercapacitor formed by placing two 100 F Bussmann PowerStor supercapacitors [78] in series (the supercapacitor size is varied in the second experiment). The supercapacitor is interfaced to a DC-DC converter based on the LT1615 [89]. The DC-DC converter is configured to accept input voltage levels of 1.5 V to 3.0 V, and to output a constant 3.3 V, which is used to power the WSN node. Note that unlike the previous chapters where we used Cymbet Enerchips as energy storage devices, we utilize supercapacitors in these experiments. The change in hardware enables us to run experiments involving changes in hardware parameters, specifically the capacity of the energy storage device. The Cymbet Enerchips that we utilized in previous chapters come in an evaluation board form factor and thus cannot have their capacities changed. In contrast, we can easily change the capacity of an energy storage device based on supercapacitors by connecting the component supercapacitors in series or parallel. The nodes utilized in this study are TelosB nodes running TinyOS. The application running on the node comprises of a simple sleep-send loop, with the send performed through LowPowerListening [104]. Senders are configured to assume the wakeup interval of the receiver as 1-s. All sends are also configured as broadcasts - therefore, at each send, the radio of the node remains active for 1-s before returning to sleep. The interval between the sends is a parameter that is varied in the first experiment. The packets from the node are received by a base station which consists of another TelosB node connected via a USB cable to a netbook. The base station keeps track of packets' arrival times, and logs the contents of the packets received. The packet has a payload

of a single variable which is incremented by the sender node prior to each send. The variable enables us to verify that the node is in continuous operation, and that it has not restarted. The physical setup of the test apparatus is the same as that in Chapter 4.5. The three simulated days in Chapter 4.5 are also used in the two experiments, but with some modifications. Instead of starting the sequence at midnight (as is the case in the previous experiments), we start the sequence at *sunrise*. Each experiment still runs for 24 hours, therefore the end of each sequence corresponds very closely to the sunrise of the next day. The majority of the experiments involve the measurement of the supercapacitor voltage. For this we utilize a Keithley 2401 sourcemeter connected to a PC running NI Labview. The Labview program takes in a voltage reading every minute.

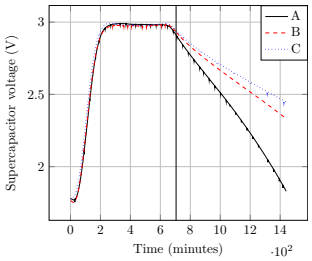
4.7.1 Experiment 1: Send interval

In the first experiment, we test how the node supercapacitor will evolve for different values of the send interval at different times of the year. We monitor the supercapacitor voltage as it is proportional to the energy stored in the capacitor. This experiment is an example of how *algorithm parameters* can be tested with the new test methodology. We test three send intervals: 20 seconds, 40 seconds, and 60 seconds; we denote the setups with these send intervals as Setups A, B, and C, respectively. The initial supercapacitor voltage is set to 1.767 V. The results for the summer solstice, autumnal equinox, and winter solstice are shown in Figure 4.16a, Figure 4.16b, and Figure 4.16c, respectively. The plots also show the time for the sunset, indicated by a single black vertical line.

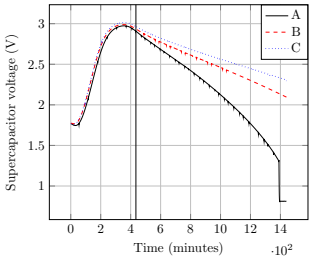
From the results it is apparent that a shorter send interval translates to greater energy consumption: this is especially pronounced during nighttime (right of the vertical line) where we see the voltage for Setup A decreasing faster than then voltage for Setup B, which in turn, decreases faster than the voltage for Setup C. It is also apparent from the figures that there is a peak level to which the supercapacitor voltages can rise. In Figure 4.16a and Figure 4.16b, where



(a) Summer solstice



(b) Autumnal equinox



(c) Winter solstice

Figure 4.16: Results for the first experiment. Send interval varied: Setup A, 20-second send interval; Setup B, 40-second send interval; Setup C, 60-second send interval. Initial voltage 2.5 V. 50 F supercapacitor.

there are numerous daylight hours, we see the voltages for all setups remaining at the peak level for extended periods of time. In contrast, in Figure 4.16c the voltage of some setups barely reach the peak before starting to decrease.

In Figure 4.16a it can be seen that for all setups, the final supercapacitor voltage is higher than the initial. This increase in supercapacitor voltage after a 24-hour period indicates an energy *surplus* - more energy was harvested than was spent during the day. As discussed in [59], to ensure *energy neutral operation*, a daily breakeven or surplus in energy is required: energy neutral operation is the state of an energy harvesting-powered system where it consumes only an amount of energy equivalent to or less than that which it regularly harvests, thus resulting in potentially perpetual operation. Dynamic power management algorithms can use the increase in voltage level as a cue that the send interval (duty cycle) can be decreased (increased).

The energy surplus for all setups remain during the autumnal equinox (Figure 4.16b). However, the differences between the final and the initial voltage levels decreases.

This is to be expected, as the shorter charging periods (the time between sunrise and sunset) lead to less energy being harvested compared to the summer solstice.

The surplus for Setups B and C persists even in the winter solstice (Figure 4.16c), although the differences between the final and initial voltages become smaller than those during the autumnal equinox. As for Setup A, the final voltage level is lower than the initial level, indicating an energy *deficit*: a deficit happens when more energy is consumed than harvested during the 24-hour period. A deficit implies that the send interval is not sustainable if consecutive days will display the same pattern, since the energy stored in the energy storage device will progressively get lower. Our experiment results suggest that a send interval of 20-s is too short for our system, especially if the goal is to incur a daily energy surplus or breakeven. Dynamic power management algorithms can take the calculated deficit as a cue to increase (decrease) the send interval (duty cycle).

Setup A in Figure 4.16c actually exhibits not just a deficit but also a sudden *collapse* in the supercapacitor voltage which causes the node to stop operating. Continuing the simulation does *not* cause the node to resume operation indicating the inability of the design to perform *cold booting* (the process of booting up via energy harvesting while coming from a very low level of charge).

We note that some deficits are *inevitable*. Because there is a limit to the energy that a supercapacitor can store, the voltages of some setups with the same power consumption but different initial voltages can converge at the peak level. In the absence of sunlight, the voltages of these setups will decrease in a similar manner, resulting in a common or highly similar final level. We see this Figure 4.16a, with Setups C and C*. Setup C* has the send interval as Setup C, but with a higher initial voltage of 2.0 V. The range of voltages for which this convergence will happen depends on the length of the day portion with sunlight as well as the power consumption of the node. Long daylight hours give more opportunities for a wider 'band' of initial voltage values to eventually

converge. A low node power consumption makes it easier to reach the peak voltage level, therefore also resulting in a wider band. The final voltage level depends on the node power consumption and number of hours without sunlight. A high enough power consumption and prolonged night could possibly result in a final voltage that is *lower* than any value in the initial voltage band, resulting in an inevitable deficit. In such a situation, the deficit can only be prevented by decreasing the node power consumption, not by increasing the initial voltage level. Alternatively, it is also possible for the final voltage level to fall *within* the values in the initial voltage band. In such a situation, those with initial voltage levels higher than the said final level will have an inevitable deficit.

4.7.2 Experiment 2: Supercapacitor size

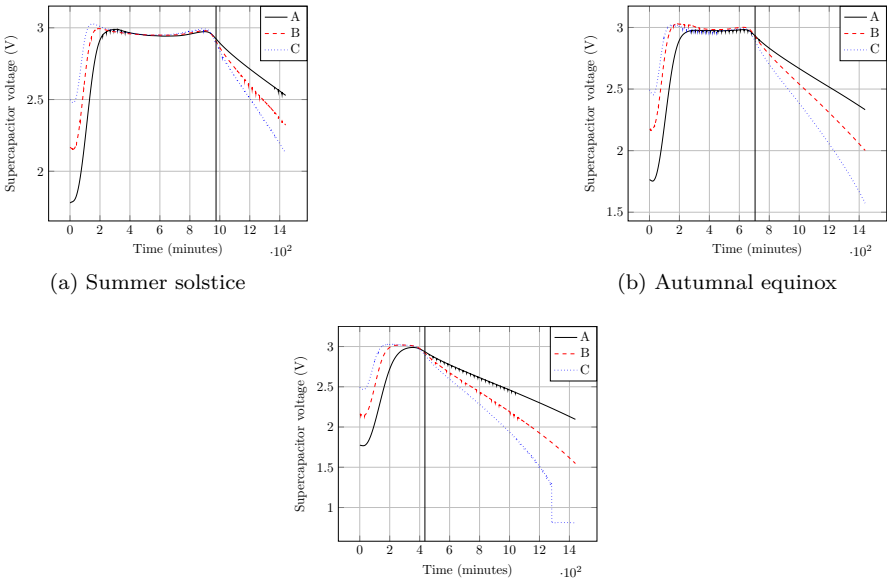
In the second experiment, we test how the supercapacitor voltage will evolve under different supercapacitor sizes. This is an example of how *hardware parameters* can be tested with the test methodology. The node utilized has a send interval of 40 seconds. We test three supercapacitor sizes: 50 F, 33.33 F, and 25 F. We denote the setups with these supercapacitors as Setups A, B, and C, respectively. All supercapacitors are formed by placing two or more 100 F supercapacitors in series.

The energy stored in a capacitor is

$$\mathsf{E} = \frac{1}{2} \times \mathsf{C} \times \mathsf{V}^2 \tag{4.10}$$

where C is the size of the capacitor and V is its voltage. For the supercapacitors to have the same initial energy levels, they must have different initial voltages. We choose the values 1.767 V for Setup A, 2.165 V for Setup B, and 2.5 V for Setup C. The initial energy is chosen so that all supercapacitors have initial voltages higher than 1.5 V but lower than 3.0 V, the designed operational limits for the DC-DC converter. The results for the summer solstice, autumnal equinox, and winter solstice are documented in Figure 4.17a, Figure 4.17b, and

Figure 4.17c, respectively.



(a) Summer solstice

(b) Autumnal equinox

(c) Winter solstice

Figure 4.17: Plots for the second experiment. Supercapacitor size varied: Setup A, 50 F; Setup B, 33.33 F; Setup C, 25 F. 40-second send interval.

It can be seen in the figures that most setups reach the same peak voltage of around 3.0 V. During days with long charging periods (daylight hours), the supercapacitor voltages remain for an extended period of time at the peak level. An exception is Setup A during the winter solstice (Figure 4.17c) whose supercapacitor voltage never reaches 3.0 V before starting to decrease. The presence of a common peak voltage has the consequence that the supercapacitors are able to store different amounts of energy.

To ensure survivability, the energy storage device must be large enough to store sufficient energy for system operation through the hours without sunlight. From our experiments it is apparent that a 25 F supercapacitor is insufficient for continuous year-long operation: Setup C ends up with a deficit in each day pattern tested, and its supercapacitor voltage collapses during the winter solstice (Figure 4.17c). We note that increasing the initial voltage will *not* change the situation since the peak voltage level is already reached by Setup C during the

winter solstice. If we assume for simplicity that the peak voltage is attained right before the decrease in voltage due to lack of sunlight, we calculate that Setups A and B consume 113.53 J and 112.01 J respectively during the winter solstice evening. In comparison, Setups C's peak voltage translates to 114.6 J of stored energy. Given the very small margin, we can deduce that 25 F is too small to store the energy required by the system to operate continuously during the winter solstice evening. We also note that at the point of voltage collapse, Setup C theoretically still has 20.48 J left in stored energy. The collapse is a system behaviour that will be missed by simulations that employ simplistic models of the energy storage device.

Setup B ends up with a surplus during the summer solstice (Figure 4.17a) and deficits during the autumnal equinox (Figure 4.17b) and winter solstice (Figure 4.17c).

Setup A ends up with a surplus in each of the day patterns tested - this indicates that a 50 F supercapacitor *can* store enough energy that will sustain the system through any night of the year: whether it *will* store enough energy is dependent on the energy it has at the beginning of the day, and the time of year.

Aside from capacity, operational requirements of other components must also be taken into account when choosing or sizing an energy storage device. For example, it is possible that a large supercapacitor will have the capacity to store enough energy that will power the system through the night but its terminal voltage will fall below the operational limits of the DC-DC converter. This will still result in the system shutting down, as the DC-DC converter is sensitive to the voltage in its input terminals, not directly to the energy stored in the device generating the said voltage.

4.8 Limitations

The primary limitation of the documented test methodology is that the tests it employs are *approximations* of actual day patterns. Because of weather effects, the energy that can be harvested at a specific day, time and place is inherently stochastic. This said, the test methodology does provide researchers and hardware/software designers well-defined limits of the energy conditions a node (or network of nodes) may encounter and enables repeatable experiments under these conditions.

This test methodology should not be used to compare different solar panels. It can however be used for testing (and sizing) different components 'downstream' of the solar panel such as the DC-DC converter, the energy storage device, and the microcontroller itself.

A limitation of our current test apparatus is the size of the solar panel with which it can be used. All parts of the solar panel must be exposed to the same level of brightness, and the size of the panel to which this can accurately be applied depends on the light source. Our test apparatus, which utilizes an 8-LED array, has been employed on solar panels sized 30 mm x 30 mm. We have also been constrained by the size of the solar panel which can be tested by our solar simulator (which we use for deriving the I_{SC}). Nevertheless, larger light sources can be used, which will be capable of accommodating larger solar panels. The only remaining issue then will be the derivation of the I_{SC} of the solar panel, as most solar simulators are only designed for testing solar cells. Some solar panels come with datasheets stating the I_{SC}. Alternatively, the panel I_{SC} can also be derived from the cell I_{SC} provided that the circuit arrangement (series, parallel) of the cells comprising the panel is known.

A further limitation of the current test methodology and test apparatus is that they do not take into account the solar panel's temperature. The temperature of a solar panel can have an effect on its performance. A test apparatus which takes into account and controls the temperature of the solar panel can be

constructed by incorporating features of TempLab [15]. TempLab is an extension for WSN testbeds that allows the control of node temperatures by using infra-red light bulbs and Peltier cooling modules.

4.9 Related Work

To the best of our knowledge, there are no other studies concerning the indoor testing of solar-powered WSNs without the use of solar simulators. We therefore focus on the test methodologies and power models used in studies concerning solar-powered or energy-harvesting WSNs. We divide such test methodologies into two categories: those that utilize software-based simulations and models (Chapter 4.9.1), and those that are used with real-world systems (Chapter 4.9.2).

4.9.1 Software-based simulations and models

Numerical simulations

Most studies on power management algorithms that rely on scheduling tasks (such as [112], [65], and [19]) test their algorithms using numerical simulations with very simple power models. The simulators utilized in these studies are mostly custom-made, and implemented in C or MatLab. The utilization of custom simulators makes experimental comparison and verification difficult. A more significant problem with the test methodology employed in these studies is their sheer simplicity. Hardware non-idealities and the environment are rarely taken into account, therefore, significant work and verification using a methodology suitable for actual devices is required before such algorithms are feasible for actual systems or devices.

Discrete event simulator extensions

An additional approach to simulating the power consumption of energy-harvesting nodes utilizes extensions to existing simulation frameworks. Notable examples

of this approach are SensorSim [119] [120], sQualNet [156], and PowerTOSSIM [136]. PowerTOSSIM [136] is an extension to TOSSIM [64] which enables the simulation and measurement of the energy consumption of TinyOS applications. When running a simulation, PowerTOSSIM analyses the instructions in the program and depending on the components utilized by the instruction (and the states the components are in), computes the power consumption of the node.

The accuracy of the power values in these simulators is usually improved by using values derived from hardware profiling. In PowerTOSSIM for example, a hardware power consumption model is built using microbenchmarks that exercise each hardware component independently. Power consumption readings are taken while microbenchmarks are running in the node. Other tools for power metering of wireless sensor nodes include Quanto [37] and SPOT [56].

The main drawback of this approach is the difficulty of carrying out hardware profiling, and in the case of TOSSIM and PowerTOSSIM, creating an accurate instruction-component model. TinyOS supports several WSN node platforms (Mica [48], Mica2, MicaZ, and TelosB, to mention but a few), but TOSSIM and PowerTOSSIM currently only support the MicaZ platform. An additional drawback is that these simulator extensions only consider the node itself, and not the components usually added to node platforms to enable them to harvest energy. PowerTOSSIM for instance, does not take into account DC-DC converter efficiency, leakage in the energy storage device, or the environment (sunlight patterns).

Summary

All the above approaches are based on software-based simulations or models, and are applicable to specific types of studies. For instance, for initial investigations of power management algorithms, it is understandable that they will be implemented using custom simulators and that power models will be quite simplistic. Nevertheless, we assume that work in WSNs, regardless how initially theoretical in nature, are carried out with the long term view or expectation

that they will be implemented in actual, physical systems. It is in the testing of these systems that we make a contribution with this study. Currently-used methodologies for testing actual physical systems are discussed next.

4.9.2 Test methodologies for actual physical systems

Most systems destined for actual deployments are not tested indoors prior to deployment. Instead the energy consumption of the node (in joules per operation, for instance) and the energy that can be harvested from the environment are estimated, and the parameters (such as the application duty cycle) are subsequently set. This approach has obvious drawbacks: the performance (and survivability) of the system will depend on the accuracy of the estimates, and such estimates are difficult to acquire with high precision because of factors such as hardware non-idealities (the efficiency of the DC-DC converter or non-linearity of the energy storage device's charging function, for example) and the inherently stochastic nature of the energy that can be harvested from the environment. To compensate, very conservative parameters are often used, but this leads to loss of performance and will still not be a hard guarantee of system survivability.

One such study which employed the 'estimate-and-deploy' methodology is described in [149]. They describe an approach used in designing the energy harvesting subsystem of sensor nodes that were used in studying hydrological cycles in forest watersheds. The computations underlying the choice and sizing of components are explained, as well as the rationale for key design decisions. The application that was run on the node is not adaptive: it operated at the same duty cycle, regardless of the environmental conditions. While best effort was made in estimating the parameters, nodes still shut down during the deployment.

The duty cycling scheme presented in [63] was tested on actual hardware (TI eZ430-RF2500-SEH [153]) with the solar panels being illuminated by a desktop lamp. However, the brightness of the lamp was not calibrated in any way, nor

was it cycled or modulated to reflect an actual day pattern. Given the lack of calibration, it is very much possible that the energy harvesting system was induced to produce energy higher than it will under actual outdoor conditions. As we highlighted in Chapter 4.5, this is a common problem that will lead to poorly-developed systems.

A significant departure from the methods used in most other work, [55] performed experiments (empirical comparison between two different energy harvesting WSN node designs) outdoors under natural sunlight: the experiments were done during 'sunny days in mid-October' to minimize weather effects. The experiments performed, while valid, are difficult (if not impossible) to replicate and do not capture seasonal effects.

A drawback with the reliance on actual deployments for testing is that no two day patterns are exactly the same: because of this lack of repeatability, parameters, algorithms, and designs are difficult to objectively compare. Even if the variability between day patterns is ignored, as in [55], relying on actual deployments means being restricted by one's location and the current season. For example, to test how the system will perform during winter, the study would have to be performed in winter. A system destined for deployment in sub-Saharan Africa will have to be tested, at all stages, in sub-Saharan Africa.

Our test methodology, in comparison, enables the repeatable testing of parameters, algorithms and hardware designs. The studies that result are not limited by the current season or the location where the study is being conducted. How a node (or even network of nodes) will fare in sub-Saharan Africa during winter can be studied in a controlled laboratory experiment.

4.10 Summary and future work

We present an indoor test methodology for solar-powered WSNs. The methodology is based on astronomical models and PV cell design principles, and it enables repeatable experiments without the need for expensive solar simulators.

We detail the design of a generic test apparatus which can be used in implementing the test methodology. We also detail the design of our own implemented test apparatus, which has two variants: centralized and distributed. The centralized version can be used in experiments involving isolated solar-powered WSN nodes. The distributed version can be used in testing networks of such nodes. Our implemented designs differ from the generic design in that they do not rely on a feedback loop as a control mechanism, relying instead on the stability of the light source and a modelling function specifying the relationship between the I_{SC} and the lighting element setting. The omission of the feedback loop significantly simplifies the apparatus design.

Our experiments show that the implemented test apparatuses can replicate target-theoretical I_{SC} sequences accurately, although the accuracy varies depending on the specific I_{SC} value involved. The variations in accuracy between different I_{SC} values stem from the light source's imperfect stability, as well as inaccuracies in the modelling function.

We also perform a series of experiments demonstrating how the test methodology can be used in deriving software and hardware parameters for a WSN node. The results of our experiments largely confirm and conform with well-known power management principles. We note however that prior to this work deriving such parameters through repeatable experimentation has been impossible (at least not without a solar simulator).

The study we present is primarily an *enabling* study, and can be applied in many other research areas involving actually-implemented solar-powered WSNs or WSN nodes. For example, it can be used in designing and studying dynamic power management algorithms, or testing the ability of hardware designs to perform cold booting at different times of the year.

As for improving the test methodology or apparatus, future research can proceed in several directions. Other light sources, such as Xenon arc lamps, can be tried. A test apparatus incorporating the feedback loop can also be implemented, to determine whether it will be more accurate than our implementation.

Alternatives to the astronomical model presented, such as the models in [122] and [24], can also be tried to see whether they will be more representative of outside conditions. Customized models for specific locations and times can also be created from historical meteorological data, yielding 'expected average' conditions and in the process taking into account the effects of the weather, cloud covering, etc.. Any model can theoretically be used, as long as their outputs can be converted to an I_{SC} equivalent.

CHAPTER 5

Techniques and algorithms for future capabilities: Target identification in directional sensor networks

5.1 Overview

Future energy-harvesting noise-sensing WSNs will or can have features they are not originally envisioned with. Some of these features may not even be implementable yet, or exist only in prototypical forms. Nevertheless, one way the field can advance is by anticipating and preparing for them. In this chapter, we present algorithms which enable target identification in directional sensor networks. They are relevant to noise-sensing WSNs since target identification is a highly advantageous for noise-sensing WSNs to have, and noise-sensing WSNs can be turned into directional sensors if they use directional instead of omni-directional microphones. Given that this is the first discussion of directional sensors in the context of maximizing coverage *and* target identifiability, we set aside the requirement that the WSN nodes be powered by energy harvesting. We also assume that the WSN nodes are not duty cycling; alternatively stated, we assume that the nodes are duty cycling with a duty cycle of 100%, and the sensors and the radio are always on. We reserve the case of WSN nodes executing the algorithms discussed while powered by energy harvesting as part of our possible future work. In addition to assuming that the nodes are not powered by energy harvesting we also assume significant idealities in terms of communication between the nodes. In particular we assume that the communication channel is noiseless, and that packets send between nodes are always successfully received (as long as the nodes are within communication range of each other).

This key area is admittedly the most forward-looking and speculative of the four: some features and capabilities that researchers may prepare for may simply never come to fruition. The features and capabilities of future noise-sensing WSNs will depend on the progress of related technologies (such as MEMs, energy storage devices, etc.) and while projections can be made about these technologies, there will always be an element of uncertainty about the said projections. Nevertheless, speculating and 'betting' on the possibility that enabling technologies *will* come to fruition are *still* productive exercises, as they show that there are applications for these technologies. Speculating and preparing for future capabilities can therefore spur a 'self-fulfilling prophecy', as they encourage the development of (and investment in) the technologies they are based on.

5.2 Introduction and motivation

Directional sensors are sensors whose sensing capabilities are limited within an angle range [66][67]. In comparison, an omnidirectional sensor's sensing range covers everything around it. In geometric terms, the covered area of an omnidirectional sensor is a circle centered on the sensor, while that of a directional sensor is a sector. In WSNs, nodes that are *directional* can imply that the node has directional capability in sensing and/or communication [66]. In this chapter, we solely focus on the sensing capability, and thus will interchangeably use the terms 'nodes' and 'sensors'.

Examples of sensors that are inherently directional in nature include video sensors [126], ultrasonic sensors [26], and infrared sensors [141]. Acoustic sensors (or microphones) can also potentially be directional [98], although most of the existing work in WSN literature so far have utilized omnidirectional microphones [130][43].

Because the sensed region of a directional node is constrained within an angle range, of primary concern is the *total sensing coverage* provided by such a network of directional sensors. A problem that frequently needs to be solved

is *'Given a number of directional sensors distributed in space, how should each sensor's sensing region be oriented such that the total sensing coverage of the network is maximized?'*.

Coverage-maximizing algorithms proposed in literature fall into two categories: those that are *target-centric* and those that are *area-centric*.

In target-centric algorithms, it is assumed that there are a finite number of *static* targets distributed in the area, and it is desired that as many as possible of such targets be covered. Sometimes, for issues of energy efficiency, it is also desired that the covering be also done with the least number of sensors possible - such a problem is formalized in [2] as the Maximum Coverage Minimum Sensors (*MCMS*) Problem.

In area-centric algorithms, there are no specific objects or targets of interest; the entire area is of sensing interest, and it is desired that as much of it is covered. In such algorithms, the problem is equivalent to minimizing the *overlap* between the sensed regions of sensors. An example of such an algorithm is presented in [150].

In this work, we solely focus on target-centric algorithms.

Most coverage algorithms focus on maximizing the number of targets covered. This is reasonable when the targets are continually being monitored, and easy target identification is built-in to the system. This is true to a certain extent for visual sensor networks: for instance, a Closed Circuit Television (CCTV) camera (being remotely watched by a person) monitoring a street intersection. However, this assumption does not hold true for all situations. For instance, if we assume that a single acoustic sensor is monitoring two possible sound sources, in general, the sensor will be able to detect that a sound was generated, but not which source generated it - at least not unless the sound generated by each source is distinct, and even then, not without further digital signal processing.

As a slight deviation from all previous studies, we shall work with the following assumptions

1. Targets generate *events*, and these events randomly occur. Events occur only one at a time, and they are brief enough that they do not overlap in time.

2. Events can be detected by the sensor if the source is within the sensor's sensed region, but the event themselves say nothing about which source generated it.

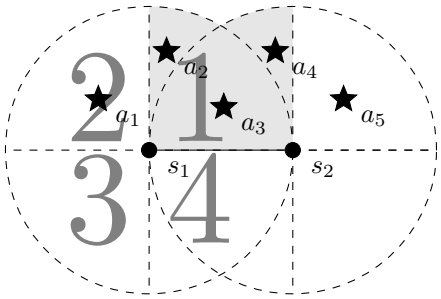


Figure 5.1: Diagram for the first example.

In this work, instead of just focusing on the number of targets covered, we will also concern ourselves with the identifiability of the targets or event sources.

As a motivating example, we present Figure 5.1, where the dots s_1 and s_2 are sensors while the stars a_1 - a_5 are the targets. The dotted regions around each sensor represent the possible covered or sensed region of each sensor. Each sensor can be oriented to be in 1 of 4 possible orientations. Orientation 1 represents the region covered by the sector from 0° to 90°, 2 covers 90° to 180°, 3 covers 180° to 270°, and 4 covers 270° to 360°. In this example, the boundaries of the sensed regions of each sensor are aligned with the cardinal directions (and with each other). The 4 possible orientations for s_1 are labeled in Figure 5.1. It must be noted however, that such an alignment is not required by the algorithms that will be discussed. In Figure 5.1 s_1 is in configuration 1, while s_2 is in configuration 2. With these configurations, a_2 is covered by s_1, a_3 is covered by s_1 and s_2, and a_4 is covered by a_2. A total of 3 targets are covered. a_1 and a_5 are not covered.

It is easy to verify that 3 is the maximum number of targets that can be covered by any network configuration. A *network configuration* is a set of active sensors, each with its corresponding orientation.

However, the configuration s_1 - 1 (shorthand for s_1 in orientation 1), s_2 - 2 is not the only configuration that yields 3 covered targets. s_1 - 2, s_2 - 2 also yields 3 covered targets, as will s_1 - 1, s_2 - 1. There are differences however in the identifiabilities conferred by these configurations to the targets that they cover.

Let us begin with s_1 - 2, s_2 - 2. When a_1 generates an event, s_1 will be able to detect it, and we know for certain that a_1 generated the event since it is the only target covered by s_1. When a_3 generates an event, s_2 will be able to detect it, but we are not sure whether it was a_3 or a_4 that generated the event. The best that can be done is hazard a guess with 50% probability of being correct. The same analysis holds for s_1 - 1, s_2 - 1.

Compare this with s_1 - 1, s_2 - 2. When a_2 generates and event, s_1 will be able to detect it. At first glance, it seems like we might not be able to distinguish whether it was a_2 or a_3 which generated the event since both are covered by s_1. However, we can know that it is *not* a_3, since s_2 did not detect anything. Hence, it must be a_2 which generated the event. In other words, whether a target generated an event or not can be deduced not just from which sensors detected something, but also from those that did *not* detect anything.

We call the set of sensor states (where *state* indicates whether a sensor detected something or not) which signifies a target generating an event as the target's *syndrome*. A syndrome is a tuple of values, one for each active sensor in the system, each denoting whether a sensor will detect anything upon the target generating an event. For a sensor x, let s_x be denoted by s_x in the tuple if the sensor will detect anything, and s_x' if it will not. In our latest example, a_2 has the syndrome $s_1 s_2'$. Table 5.1 enumerates the syndrome for each covered target in each of the network configuration that yields 3 covered targets.

In Table 5.1, we can clearly see why the configuration s_1-1, s_2-2 affords

	s_1 - 1, s_2 - 2	s_1 - 2, s_2 - 2	s_1 - 1, s_2 - 1
a_1	-	$s_1 s_2'$	-
a_2	$s_1 s_2'$	-	$s_1 s_2'$
a_3	$s_1 s_2$	$s_1' s_2$	$s_1 s_2'$
a_4	$s_1' s_2$	$s_1' s_2$	-
a_5	-	-	$s_1' s_2$

Table 5.1: Configurations that yield 3 covered targets, and their resulting syndromes.

better target identifiability: in that configuration, each covered target has its own syndrome. In comparison, in the other two configurations, two targets have to share a single syndrome, resulting in ambiguity when identifying their events.

It must be noted that some aspects of Figure 5.1 do not hold true in other situations.

Firstly, sometimes, assigning a single syndrome for each target is simply impossible. Nevertheless, it is conceivable that even in such situations, it is desirable to minimize the ambiguity between events as much as possible.

Secondly, in other networks (especially those that are *underprovisioned*, meaning there are few sensors relative to targets), it is possible that improved target identifiability will come at the cost of less targets covered. The acceptable trade-off between the number of targets covered and their identifiability will vary from one application to the next.

Another use of the concept of identifiability is in *overprovisioned* networks - that is, networks where there is a surplus in the number of sensors, and even after the maximum possible number of targets has been covered, there are still sensors that are not covering anything. In previous studies, the extra sensors are used in extending the network lifetime: sensors form *cover sets* that take turns covering the targets [17]. Clearly, another possible use for such extra sensors is in increasing the identifiability of targets.

This chapter makes three contributions: firstly, the introduction of the concept of *syndromes*; secondly, the definition of the Maximum Target Identifiability-

Aware Utility with Minimum Sensors (MTIAUMS) problem; and finally, six heuristic algorithms that determine network configurations that strike a balance between identifiability and the number of targets covered.

This chapter is structured as follows. The notations utilized are discussed in Section 5.3. The problem is formally defined, and its NP-hardness proven in Section 5.4. Centralized heuristic algorithms are presented in Section 5.5, while distributed heuristic algorithms are presented in Section 5.6. The methodology used to test the algorithms, and the results of the simulations, are presented in Section 5.7. A discussion of related work follows in Section 5.8, while Section 5.9 summaries the chapter.

5.3 Notations

The notations used in this chapter are tabulated in Table 5.2. We note that the notations presented in Table 5.2 are local to this chapter, and does *not* apply to the rest of the thesis.

5.4 Problem definition

To give consideration to the fact that not all setups will be like that in Figure 5.1 (where the network configuration which maximizes the number of targets covered also maximizes the number of syndromes), we first introduce the concept of *target utility*, $\mathsf{u_i}$

$$\mathsf{u_i} = \alpha + (1 - \alpha)(\mathsf{certainty_i}) \tag{5.1}$$

$\mathsf{u_i}$ will depend on the network configuration Z. As previously mentioned, a network configuration is a set of active sensors, each with its corresponding orientation. α is a parameter (with value between 0 and 1, inclusive) defined by the user, which indicates the desired trade-off between number of targets covered and identifiability: a higher α indicates that the number of covered

Variable/Quantity	Description
M	number of targets
a_{i}	a specific target, $1 \leq \mathsf{i} \leq \mathsf{M}$
A	the set of all targets $A = \{a_1, a_2, ... \ a_{\mathsf{M}}\}$
N	number of nodes
s_{i}	a specific node, $1 \leq i \leq \mathsf{N}$
S	the set of all nodes $S = \{s_1, s_2, ... \ s_{\mathsf{N}}\}$
W	number of orientations with which each node can work with
$\phi_{\mathsf{i},\mathsf{j}}$	set of targets covered by node i when it is working with orientation j, $1 \leq \mathsf{i} \leq \mathsf{N}$, $0 \leq \mathsf{j} \leq \mathsf{W}$. Note that we allow j to take on the value of 0: this indicates that the node's sensing mechanism is inactive, thus $\phi_{\mathsf{i},0} = \{\}, \forall \ \mathsf{i}$
Φ_{i}	set of all targets within range of s_{i}, *regardless* of orientation; $\Phi_{\mathsf{i}} = \{\cup \ \phi_{\mathsf{i},\mathsf{j}} \mid 1 \leq \mathsf{j} \leq \mathsf{W}\}$
Φ'_i	set of all targets within range of s_i's one-hop neighbours, sans those also seen by s_i; $\Phi'_{\mathsf{i}} = \{\cup \ \Phi_{\mathsf{j}} \setminus \Phi_{\mathsf{i}} \mid s_{\mathsf{j}} \in Q_{\mathsf{i}}\}$
α	user-defined parameter indicating desired trade-off between number of targets covered and their identifiability; higher value indicates more preference for number of targets covered; $0 \leq \alpha \leq 1$
Z	network configuration, set of active sensors and corresponding orientation; set of (i, j) pairs, where $\mathsf{i} \in S$, $1 \leq \mathsf{j} \leq \mathsf{W}$
u_{i}	target utility for a given Z, $1 \leq \mathsf{i} \leq \mathsf{M}$
U	sum of all target utilities, for a given Z
Q_{i}	set of nodes comprised of one-hop neighbours of s_{i}

Table 5.2: Notations.

targets is more important than the number of syndromes in the system, a lower α value indicates the reverse. $\mathsf{certainty}_{\mathsf{i}}$ is the level of certainty with which a target can be identified when it generates an event. Like u_{i}, it is dependent on Z (it is the reason why u_{i} is dependent on Z). It can be defined as:

$$\mathsf{certainty}_{\mathsf{i}} = \frac{1}{\#\text{ of targets sharing the same syndrome as } a_{\mathsf{i}}} \tag{5.2}$$

Building on the concept of target utility, we define system utility, U:

$$U = \sum_{i=1}^{M} u_i \tag{5.3}$$

Another definition for U will be

$$U = \alpha(\text{number of targets covered}) + (1 - \alpha)(\text{number of syndromes in the system}) \tag{5.4}$$

The second term holds because if all the certainties of all covered targets are summed, one will end up with the number of syndromes in the system. Unless explicitly stated, references to 'utility' in this chapter must be taken to mean 'system utility'.

Our goal is to provide coverage to targets, maximizing the system utility as defined by Equation 5.4, while activating as few sensors as possible. We call this problem the Maximum Target Identifiability-Aware Utility with Minimum Sensors (MTIAUMS) problem. A more formal statement of the problem will be: *Given a set of targets A and a set of sensors S (each with W possible sensing orientations), find a network configuration Z (consisting of a set of active sensors, along with their corresponding orientations), such that the resulting system utility U is maximized and the cardinality of Z is minimized.*

It must be noted that the computed system utility is affected by the ratio of targets to sensors, and their densities (targets per unit area, sensors per unit area). If there are significantly more targets than sensors (several orders of magnitude higher, for instance), it is actually possible for heuristic algorithms that only aim to maximize the number of covered targets to attain a higher system utility than our algorithm. While the algorithm will still confer better identifiability to covered targets in such a situation, that can possibly be hidden by the fact that there are significantly more possible targets that can be covered (first term, Equation 5.4) than there are possible syndromes that can be generated (second term, Equation 5.4). In such cases, it might be helpful to take the

metric $\frac{targets}{syndrome}$ into account when evaluating solutions. In this work, we will deal with cases wherein the number of targets is comparable to the number of sensors.

In this study, we propose heuristic solutions to the MTIAUMS problem. A heuristic solution is useful since the problem is NP-hard (this will be proven in the next subsection). We propose six (6) heuristic solutions to the problem. The six algorithms are firstly divided between *centralized* and *distributed* algorithm subfamilies. Centralized algorithms intuitively have the advantage of 'seeing' the problem in its entirety when computing solutions. However centralized solutions may be impractical in terms of communications, as all sensor states must either be sent to a single computing node or exchanged with all nodes in a mesh-like manner. Therefore, a distributed solution is sometimes necessary. Within each algorithm subfamily there is an algorithm (TIA-CGA for centralized; TIA-DGA for distributed) which is a 'true' MTIAUMS solving algorithm: it really takes the system utility into account at each step as it searches greedily for a solution. These algorithms however have high computational complexity. We therefore propose another type of algorithm which is less computationally complex, the 2-stage algorithm. In a 2-stage algorithm the targets are covered first and only surplus sensors are used in increasing the number of syndromes in the system. There are two well-known algorithms for finding network configurations that cover targets in a directional sensor network: CGA and CFA. Taking this into account leads to two 2-stage algorithms for each algorithm subfamily (2S-CGA and 2S-CFA for centralized; 2S-DGA and 2S-DFA for distributed).

5.4.1 NP-hardness of the problem

We prove the NP-hardness of the problem through the *'Proof by Restriction'* method [40]. When $\alpha = 1$, the MTIAUMS problem becomes the Maximum Coverage with Minimum Sensors (MCMS) problem defined in [2]. The MCMS problem therefore is a special case of the MTIAUMS problem. It is proven in [2] that the MCMS problem is NP-hard - therefore, the MTIAUMS problem is

also NP-hard.

5.5 Centralized algorithms

The first set of heuristic algorithms that is presented is that of the centralized algorithms. Syndrome counting is discussed in Section 5.5.1. Our first heuristic algorithm, TIA-CGA, is presented in Section 5.5.2. Two other centralized algorithms are discussed in Section 5.5.3.

5.5.1 Counting syndromes

The capability to count the number of syndromes associated with a specific network configuration is of primary importance to algorithms that will be presented later. To count the covered targets and to evaluate the level of identifiability afforded by a given network configuration (or set of sensor orientations), we introduce the concept of *coverage matrix*. A coverage matrix is formed from a given Z. A coverage matrix has its rows indexed by the members of A and its columns indexed by the members of S. Let $\mathsf{z}_{\mathsf{i},\mathsf{j}}$ be a member of a coverage matrix

$$\mathsf{z}_{\mathsf{i},\mathsf{j}} = \begin{cases} 1 \text{ if } \mathsf{i} \in \phi_{\mathsf{j},\mathsf{k}} \text{ s.t. } (\mathsf{j},\mathsf{k}) \in Z; \\ 0 \text{ otherwise, inc. } (\mathsf{j},\mathsf{k}) \notin Z \ \forall \ \mathsf{k},\ 1 \leq \mathsf{k} \leq \mathrm{W}. \end{cases} \tag{5.5}$$

To count the number of syndromes in a given configuration, we use Algorithm 4. One of the inputs to Algorithm 4 is an array called covered whose element is 1 if the index corresponding to the target is covered in the coverage matrix (at least one non-zero value in the corresponding row). The array covered can be easily generated along with the coverage matrix via Equation 5.5.

Algorithm 4 works by comparing the syndromes of each target in a pairwise fashion. In the coverage matrix, the syndrome of a target is represented by the values in the row corresponding to the target number. For example, target a_1's syndrome will be the concatenation of the values in row 1: $\mathsf{z}_{1,\mathsf{j}}, 0 \leq$

Algorithm 4 Syndrome counting algorithm

```
1: Inputs: coverage matrix with elements z_{i,j} (generated using Equation 5.5);
   an array called covered whose element is 1 if the index corresponding to the
   target is covered in the coverage matrix (at least one non-zero value in the
   corresponding row)
2: Output: syndromes - the number of syndromes in the system when solution
   embodied by the coverage matrix is applied
3: syndromes ← 0
4: processed(i) ← 0 ∀a_i ∈ A
5: for 1 ≤ i ≤ M do
6:     if (processed(i) == 0) && (covered(i) == 1) then
7:         syndromes ← syndromes + 1
8:         for i + 1 ≤ j ≤ M do
9:             if (processed(j) == 0) && (covered(j) == 1) then
10:                same ← 1
11:                for 1 ≤ k ≤ N do
12:                    if z_{i,k} != z_{j,k} then
13:                        same ← 0
14:                    end if
15:                end for
16:                if same == 1 then
17:                    processed(j) ← 1
18:                end if
19:            end if
20:        end for
21:        processed(i) ← 1
22:    end if
23: end for
```

$j \leq N$. Each target, unless already 'processed', becomes a reference at some point in the algorithm. The algorithm chooses the reference row sequentially (Algorithm 4, Line 5). The reference row is then compared with rows with higher numbers (Algorithm 4, Line 8). The actual element-by-element comparison of the syndromes happens in Algorithm 4 Lines 11-15. If a row has exactly the same values as the reference row, it is marked as already processed (Algorithm 4, Line 21) and loses the chance to become a reference row itself. It will also not be eligible for comparison with any other future reference rows. The successful selection of a new reference row effectively represents the 'discovery' of a new syndrome and the syndromes variable is incremented by 1 (Algorithm 4, Line 7).

The maximum number of target-target comparisons that can possibly be

performed in the process of counting syndromes is equivalent to the number of pairwise combination of targets (Algorithm 4, Lines 5 and 8), or $\frac{\mathsf{M!}}{2!(\mathsf{M}-2)!}$. Within a target-target comparison, it is checked whether or not both targets are covered by each sensor (N, Algorithm 4, Line 11). Therefore, the computational complexity of Algorithm 4 is $\frac{\mathsf{M!}}{2!(\mathsf{M}-2)!}$

Algorithm 5 Target Identifiability-Aware Centralized Greedy Algorithm

1: **Inputs:** A; S; α - weighting factor; $\phi_{\mathsf{i,j}}$s
2: **Output:** Z - network configuration, a set of (ID of active node, configuration of active node) pairs
3: $Z \leftarrow \emptyset$
4: $V \leftarrow A$ ▷ at the end, will contain all uncovered targets
5: $Y \leftarrow S$ ▷ at the end, will contain all inactive nodes
6: new_utility $\leftarrow 0$
7: **while** 1 **do**
8: old_utility $\leftarrow$ new_utility
9: **for** (i,j) s.t. $s_{\mathsf{i}} \in Y$, $0 \leq \mathsf{j} \leq \mathsf{W}$ **do**
10: $TempNodeSet \leftarrow \emptyset$
11: $TempNodeSet \leftarrow Z \cup \{(\mathsf{i}, \mathsf{j})\}$
12: TempCovMatrix $\leftarrow$ coverage matrix generated from $TempNodeSet$ ▷ generate using Equation 5.5
13: $\mathsf{Q}_{\mathsf{i,j}} \leftarrow$ `SyndromeCount`(TempCovMatrix, associated covered array) ▷ compute number of syndromes using Algorithm 4
14: $\mathsf{U}_{\mathsf{i,j}} \leftarrow (\alpha) \times (|\phi_{\mathsf{i,j}} \cap V|) + (1 - \alpha) \times (\mathsf{Q}_{\mathsf{i,j}})$
15: **end for**
16: $(\mathsf{i},\mathsf{j}) \leftarrow \arg\max_{\mathsf{s_i} \in Y, 0 \leq \mathsf{j} \leq \mathsf{W}} \mathsf{U}_{\mathsf{i,j}}$
17: new_utility $\leftarrow$ maximum $\mathsf{U}_{\mathsf{i,j}}$ determined in previous step
18: **if** new_utility - old_utility ≤ 0 **then**
19: break ▷ utility does not increase anymore, algorithm ends
20: **else**
21: $Z \leftarrow Z \cup \{(\mathsf{i},\mathsf{j})\}$
22: $V \leftarrow V \setminus \phi_{\mathsf{i,j}}$
23: $Y \leftarrow Y \setminus \{(\mathsf{s_i})\}$
24: **end if**
25: **end while**

5.5.2 Target Identifiability-Aware Centralized Greedy Algorithm

The Target Identifiability-Aware Centralized Greedy Algorithm (TIA-CGA) is a heuristic algorithm that produces network configurations that take into account both the number of targets covered and the identifiability of those targets. The

TIA-CGA is primarily derived from the Centralized Greedy Algorithm (CGA) presented in [2]. The CGA is a greedy algorithm which at each stage of the solution computation chooses the sensor-orientation pair which adds the highest number of targets to those already covered. The main difference between the TIA-CGA and the CGA is that in an iteration, instead of choosing the sensor-orientation pair which adds the most number of newly covered targets, the TIA-CGA chooses the pair which results in the greatest additional system utility to the system. To do this, for each remaining unchosen sensor-orientation pair, it computes the number of additional targets that will be covered if the pair is chosen next, and the number of syndromes that the system will have (Algorithm 5, Lines 10-13). These two values are weighted by the priority factors α and 1 - α, respectively, and then added together (Algorithm 5, Line 14). Algorithm 5 Line 16 chooses the pair which adds the greatest system utility to the system. The algorithm will stop when the best pair found no longer improves the system utility (Algorithm 5, Lines 18-19); otherwise, the pair is added to the solution (Algorithm 5, Line 21), the sensor is removed from the set of sensors viable for selection in the next iteration (Algorithm 5, Line 23), and the loop begins anew.

To derive the computational complexity of Algorithm 5, it must be noted that in each of its iteration, it needs two pieces of information to choose the sensor-orientation pair: the number of additional targets each sensor-orientation pair will cover (if chosen), and the number of syndromes each sensor-orientation pair will add to the system (if chosen).

The number of targets covered can be determined in MNW steps, in keeping with the value derived in [2].

For the number of syndromes, there are NW sensor-orientation pairs to evaluate. In each evaluation, a coverage matrix is generated (Algorithm 5, Line 12, Equation 5.5). The coverage matrix has MN elements, so it can be assumed that it will take MN steps to evaluate. The coverage matrix is processed by Algorithm 4, with complexity $\frac{\mathsf{NM!}}{\mathsf{2(M-2)!}}$. Therefore, in each iteration of the

main loop of Algorithm 5, the complexity due to syndrome counting will be $\mathsf{NW}(\mathsf{MN} + \frac{\mathsf{NM!}}{2(\mathsf{M}-2)!})$.

The evaluation of the system utility and the choosing of the sensor-sensor configuration pair can be done in NW steps. There are a maximum N iterations of the main loop of Algorithm 5. Therefore, the overall complexity of Algorithm 5 is $\mathsf{N}(\mathsf{MNW} + \mathsf{NW}(\mathsf{MN} + \frac{\mathsf{NM!}}{2(\mathsf{M}-2)!}) + \mathsf{NW})$.

5.5.3 2-stage algorithms

The next algorithms that will be introduced are the 2-stage algorithms. One of these algorithms is based on the Centralized Force-based Algorithm (CFA), so a short introduction on CFA is in order.

The CFA is an alternative to CGA, proposed in [114]. Like CGA, CFA aims to produce network configurations that maximize the number of targets covered, and does so greedily. Unlike CGA however, when building the solution, it does not solely rely on the number of targets that will be covered by each sensor-sensor configuration pair. Instead, at each step of the computation, it chooses on the basis of the 'force' exerted by a sensor configuration (or direction) *on* the sensor. This force is defined as the ratio of the targets covered by a specific sensor configuration to the total number of targets covered by the sensor (Equation 5.6).

$$\mathsf{F}_{\mathsf{i,j}} = \frac{|\phi_{\mathsf{i,j}}|}{|\Phi_{\mathsf{i}}|} \tag{5.6}$$

The 2-stage algorithms basically apply CGA or CFA first to the problem and then attempt to increase the number of syndromes in the system by greedily using the sensors left unselected. When the first stage is CGA, the algorithm is called 2-stage Target Identifiability-Aware Centralized Greedy Algorithm (2S-CGA), and when the first stage is CFA, the algorithm is called 2-stage Target Identifiability-Aware Centralized Force-based Algorithm (2S-CFA).

In Algorithm 6, the first stage can be found in Lines 7-22. As previously

Algorithm 6 2-stage Target Identifiability-Aware Greedy Algorithm

Inputs: A; S; $\phi_{i,j}$s; CFAStage1 - binary variable, **1** if desired first stage is CFA, **0** if CGA
Output: Z - network configuration, a set of (ID of active node, configuration of active node) pairs
$Z \leftarrow \emptyset$
$V \leftarrow A$ ▷ at the end, will contain all uncovered targets
$Y \leftarrow S$ ▷ at the end, will contain all inactive nodes
new_syndrome_count $\leftarrow 0$
while 1 **do**
  if CFAStage1 == 1 **then** ▷ First stage is CFA
    Compute $F_{i,j} = \frac{|\phi_{i,j} \cap V|}{|\Phi_i \cap V|}$ $\forall$ $s_{s_i} \in Y$, $0 \leq j \leq W$
    $(i, j) \leftarrow \arg\max_{s_i \in Y, 0 \leq j \leq W} F_{i,j}$
  else ▷ First stage is CGA
    Compute $|\phi_{i,j} \cap V|$ $\forall$ $s_i \in Y$, $0 \leq j \leq W$
    $(i, j) \leftarrow \arg\max_{i \in Y, 0 \leq j \leq W} |\phi_{i,j} \cap V|$
  end if
  if $|\phi_{i,j} \cap V| == 0$ **then**
    break ▷ nothing new can be covered anymore, stage 1 ends
  else
    $Z \leftarrow Z \cup \{(i, j)\}$
    $V \leftarrow V \setminus \phi_{i,j}$
    $Y \leftarrow Y \setminus \{(s_i)\}$
  end if
end while
while 1 **do**
  old_syndrome_count $\leftarrow$ new_syndrome_count
  for (i, j) s.t. $s_i \in Y$, $0 \leq j \leq W$ **do**
    $TempNodeSet \leftarrow \emptyset$
    $TempNodeSet \leftarrow Z \cup \{(i, j)\}$
    TempCovMatrix $\leftarrow$ coverage matrix generated from $TempNodeSet$ ▷ generate using Equation 5.5
    $Q_{i,j} \leftarrow$ SyndromeCount(TempCovMatrix, associated covered array) ▷ compute number of syndromes using Algorithm 4
  end for
  $(i, j) \leftarrow \arg\max_{s_i \in Y, 0 \leq j \leq W} Q_{i,j}$
  new_syndrome_count $\leftarrow$ maximum $Q_{i,j}$ determined in previous step
  if new_syndrome_count - old_syndrome_count ≤ 0 **then**
    break ▷ nothing can be covered that will increase the number of syndromes anymore, stage 2 and algorithm ends
  else
    $Z \leftarrow Z \cup \{(i, j)\}$
    $Y \leftarrow Y \setminus \{(s_i)\}$
  end if
end while

mentioned, the first stage can either be CGA or CFA. In the interest of saving space, Algorithm 6 is made to be capable of using both CGA and CFA, with

the choice of which algorithm to use now dependent on the binary variable CFAStage1, which is assumed to be an algorithm input. CFAStage1 having the value of **1** denotes that CFA should be used (Algorithm 6, Lines 9-10), while CFAStage1 having the value of **0** denotes that CGA should be used (Algorithm 6, Lines 12-13). We call this algorithm, which represents both 2-stage algorithms, the *2-stage Target Identifiability-Aware Greedy Algorithm.*

The second stage can be found in Algorithm 6 Lines 23-39. In each iteration of the loop, the number of additional syndromes that can possibly be added by each remaining sensor-orientation pair is computed (Algorithm 6, Lines 25-30). If the most that can be added by any sensor-orientation pair is 0 (or negative), the stage and the algorithm will end (Algorithm 6, Lines 33-34); otherwise, the sensor-orientation pair is added to the solution (Algorithm 6, Line 36), the sensor removed from the set of viable/inactive sensors (Algorithm 6, Line 37), and the loop begins anew.

It must be noted that strictly speaking, the two 2-stage algorithms are not exact alternatives to the TIA-CGA. TIA-CGA always aims to maximize the system utility, thus taking into account both targets covered and syndromes generated at each step. The 2-stage algorithms only take the syndromes into account after all targets that can possibly be covered are covered. In situations where all sensors are utilized for covering targets (no leftovers) - the 2-stage algorithms degenerate into CGA or CFA (depending on which version is being used).

The complexity of the first stage of Algorithm 6 follows that of CGA or CFA: $\mathsf{N}_1(\mathsf{M}\mathsf{N}_1\mathsf{W} + \mathsf{N}_1\mathsf{W})$. N_1 is the number of sensors that will be chosen by the first step. If all sensors are utilized in covering targets (no leftovers), $\mathsf{N}_1 = \mathsf{N}$, and the complexity of Algorithm 6 becomes the same as that of CGA or CFA.

The complexity of the second stage is $\mathsf{N}_2(\mathsf{N}_2\mathsf{W}(\mathsf{M}\mathsf{N}_2 + \frac{\mathsf{N}_2\mathsf{M}!}{2(\mathsf{M}-2)!}) + \mathsf{N}_2\mathsf{W})$. This is the same as that of Algorithm 5, but without the component for counting covered targets. N_2 is the number of unselected sensors left after the first stage. The sum of N_1 and N_2 cannot exceed N (i.e., $\mathsf{N}_1 + \mathsf{N}_2 \leq \mathsf{N}$). The complexity of

Algorithm 6 is then $N_1(MN_1W + N_1W) + N_2(N_2W(MN_2 + \frac{N_2M!}{2(M-2)!}) + N_2W)$, $N_1 + N_2 \leq N$. The majority of the complexity of Algorithm 5 stems from the computation necessary to count syndromes - by sparing some (if not most) sensors from such a step (by splitting N into N_1 and N_2), the two 2-stage algorithms usually end up with shorter runtime than Algorithm 5. This will be empirically verified in Section 5.7.

5.6 Distributed algorithms

Communication costs (in terms of energy and time) will make the transmission of data between the nodes and the base station (which will run the algorithm in a centralized solution) prohibitive as the network grows in size - hence, a distributed solution is at times more practical than a centralized one. Relevant data structures to the distributed algorithms will first be discussed in Section 5.6.1. Our first distributed algorithm, 2S-TIA-DGA, is introduced in Section 5.6.2 and its operation discussed in Section 5.6.3. A comparison with two algorithms for solving MCMS will be given in Section 5.6.4, and its time complexity is discussed in Section 5.6.5. 2-stage heuristic algorithms are introduced and discussed in Section 5.6.6.

5.6.1 Data structures

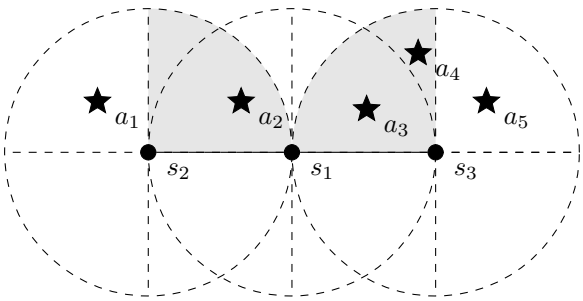


Figure 5.2: Diagram for the fourth example.

We first introduce two data structures on which the algorithm will rely on.

The data structures can be found in each node (as each node will run the algorithm).

The first data structure is the seen_targets matrix. The seen_targets matrix has its rows indexed by the members of Φ_{i}, and its columns indexed by the members of Q_{i}. Note that i in this context is the ID of the node running the algorithm. A member of the seen_targets matrix (let us call it $\mathsf{z}_{\mathsf{j,k}}$) is defined by

$$\mathsf{z}_{\mathsf{j,k}} = \begin{cases} 1 \text{ if } s_{\mathsf{k}}\text{'s current orientation covers } a_{\mathsf{j}}; \\ 0 \text{ otherwise.} \end{cases} \tag{5.7}$$

Assume that we have the setup in Figure 5.2, and that we are taking the point of view of s_1 (which has not yet decided on its orientation). Also assume that s_1 has a lower priority than either s_2 or s_3 (priorities will be discussed later) and that it has just received a protocol message from each of the nodes, informing it of their chosen orientations. The seen_targets matrix of s_1 will then be:

		Q_1	
		s_2	s_3
Φ_1	a_2	1	0
	a_3	0	1

The second data structure is the unseen_targets matrix. Like the seen_targets matrix, it has its columns indexed by Q_i. The rows on the other hand, are indexed by Φ'_i. The rows can be built dynamically (added as the node receives messages from its neighbours), or be built at an initial information exchange stage. The members of the unseen_targets matrix are defined in the same way as the members of the seen_targets matrix (Equation 5.7). Continuing with our example, if we take the point of view of s_1, the unseen_targets matrix will then be:

		Q_1	
		s_2	s_3
Φ'_1	a_1	0	0
	a_4	0	1
	a_5	0	0

5.6.2 Algorithm overview

The algorithm Target Identifiability-Aware Distributed Greedy Algorithm (TIA-DGA) is shown in Algorithm 7. In TIA-DGA, each node chooses the orientation that has the highest total system utility and then announces its choice to its neighbouring nodes. To avoid double counting, a node will only count a certain target if it has not been covered yet by a node with higher priority. An order of priority between the nodes (or at least within neighbouring nodes) is necessary to ensure that the distributed algorithm will eventually terminate [2]. Priorities in TIA-DGA are assigned by the number of targets that a node can see, or $|\Phi_i|$. The idea behind this is that nodes that can cover a large number of targets (and thus, have higher priorities) will take care of covering targets while nodes that do not have as many will then increase the number of syndromes. Ties are broken using the node ID numbers (which are assumed to be unique throughout the network).

5.6.3 Algorithm operation

Upon being initialized, a node will set its priority to the number of targets that it covers in all orientations (Algorithm 7, Line 3). It will then proceed to execute the `SetAndAnnounceBestOrientation` function (Algorithm 8).

For each of the possible orientation, the number of targets that can possibly be acquired can be determined by counting the number of relevant rows in the seen_targets matrix with all-zero entries (Algorithm 8, Line 3): because the relevant rows are those whose index belong to $\phi_{\mathrm{i,j}}$, it is representative of a target that can be seen by the sensor; the row having all-zero entries signifies that it

Algorithm 7 Target Identifiability-Aware Distributed Greedy Algorithm

Inputs: α; $\phi_{i,j}$s; W; Q_i; Φ_i; Φ'_i;
Output: j_chosen - chosen orientation for the sensor node
Set priority $PR_i = |\Phi_i|$
`SetAndAnnounceBestOrientation`();
while a protocol message is received from sensor n $\in Q_i$ s.t. $PR_n \geq PR_i$ **do**
    if $PR_n == PR_i$ **then**
        if $n > i$ **then**
            continue;
        else
            stop processing message, throw it away
        end if
    end if
    Based on the content of the message and Equation 5.7, update seen_targets matrix and unseen_targets matrix
    `SetAndAnnounceBestOrientation`();
end while

Algorithm 8 Algorithm for setting and announcing best orientation (`SetAndAnnounceBestOrientation`)

j_old $\leftarrow$ j_chosen
for j s.t. $1 \leq j \leq W$ **do**
    $P_j \leftarrow$ number of rows in seen_targets matrix with all 0s, s.t. target corresponding to the row $\in \phi_{i,j}$
    $R_j \leftarrow$ number of *unique* rows in seen_targets matrix that can also be found in unseen_targets matrix, s.t. target corresponding to the seen_targets matrix row $\in \phi_{i,j}$
    if P_j != 0 **then**
        $R_j \leftarrow R_j + 1$
    end if
    $U_j \leftarrow (\alpha) \times (P_j) + (1 - \alpha) \times (R_j)$
end for
if $\arg\max_{1 \leq j \leq W} U_j == 0$ **then**
    j_chosen $\leftarrow 0$
else
    j_chosen $\leftarrow \arg\max_{1 \leq j \leq W} U_j$
end if
if j_chosen != j_old **then**
    Set orientation to j_chosen
    Send a protocol message including priority and ϕ_{i,j_chosen}
end if

has not been covered yet by any other sensor. The algorithm will then determine how many additional syndromes it can create by choosing the orientation under consideration. This can be done by counting the unique row patterns (we define *pattern* here as the concatenation of the column-by-column values)

in the seen_targets matrix that have at least one equivalent row pattern in the unseen_targets matrix (Algorithm 8, Line 4). The idea behind this is that a new syndrome is actually assigned to a target whenever a node chooses to cover it. However, it is possible that no new syndrome is added to the system as the assigned syndrome can simply be a *redefinition* of the previous syndrome. The *addition* of a new syndrome to the system will only happen if and only if

1. There are other targets that share the target's current syndrome, and
2. Those targets will *not* share the new syndrome that will be assigned.

Going back to the previous example, should s_1 choose orientation 1, it will add a new syndrome to the system by choosing orientation 1: a_3 will be given a syndrome different from that of a_4, which it used to share the same syndrome with. Should s_1 choose orientation 2, the syndrome of a_2 will be *redefined*, but no new syndrome will be added to the system, since only a_2 uses the old syndrome.

After the number of possibly newly covered targets and possibly newly added syndromes is counted, they are then weighted and added, resulting in the utility (Algorithm 8, Line 8) for the orientation. This is done for each of the possible orientations.

If there is no utility that can be had from choosing any orientation, the node will turn the sensing mechanism off (Algorithm 8, Line 11). It is assumed however, that even with the sensing mechanism turned off, the node can still send and receive protocol messages. If there is at least one orientation with a non-zero additional utility, the orientation with the highest additional utility is chosen (Algorithm 8, Line 13), and the orientation set to it (Algorithm 8, Line 16). The node will then broadcast a protocol message containing its priority, and the targets that it has chosen to cover with its orientation (Algorithm 8, Line 17). To minimize congestion and save energy, a protocol message is only sent if the node changed its chosen orientation (Algorithm 8, Line 15). Take note that the protocol message will contain $\phi_{i,j}$, meaning, it contains *all* targets

covered by the chosen orientation, even those already covered by another node.

Assuming that the node is still in its initialization stage, the matrices seen_targets and unseen_targets will not contain anything (basically, this is the same as the node assuming that all of its neighbours are turned off).

After the initialization stage, the node will begin receiving messages from its neighbours. After determining that a message should be processed (i.e., sender has higher priority than the node), the node will update the matrices seen_targets and unseen_targets with the contents of the message (Algorithm 7, Line 13), and the function `SetAndAnnounceBestOrientation` is executed.

5.6.4 Comparison with DGA and DFA

We compare our algorithm with the Distributed Greedy Algorithm (DGA) [2] and the Distributed Force-based Algorithm (DFA) [114], two distributed heuristic algorithms designed to maximize the number of targets covered by a network of directional sensors.

In DGA, the priorities are randomly assigned, and the nodes choose the orientation with the highest number of targets not covered by neighbouring nodes with higher priorities.

In DFA, the priorities are determined by the highest force the node experiences in any orientation, with the force in a given orientation defined by Equation 5.6.

If the force value is the same with a neighbouring node, the total number of targets covered are then compared; if that are also equal, node ID values are used to ultimately break the tie. Like in DGA, the nodes choose the orientation with the highest number of targets not covered by neighbouring nodes with higher priorities.

5.6.5 Time complexity

Similar to DGA (and DFA), nodes in TIA-DGA have definite priorities, ensuring termination in finite time. Another implication of this is that it also shares the

two algorithms' time complexity in the worst case (which happens when sensors reach their final states one-by-one in the order of their priority), which is $O(n^2)$ [2].

5.6.6 2-stage distributed algorithms

We also introduce 2-stage algorithms for solving the MTIAUMS problem in a distributed way. Similar to their centralized counterparts, the idea behind the distributed 2-stage algorithms is to cover the targets first using distributed heuristic algorithms designed for solving the MCMS problem (first stage). The number of syndromes is then increased using the remaining sensors (second stage). We introduce two 2-stage algorithms: 2-stage Distributed Greedy Algorithm (2S-DGA) and 2-stage Distributed Force-based Algorithm (2S-DFA). The difference between the two lies in the MCMS-solving algorithm used in the first stage: 2S-DGA uses DGA for its first stage, while 2S-DFA uses DFA for its first stage. An algorithm both representing 2S-DGA and 2S-DFA is presented in Algorithm 9.

The primary difference between DGA and DFA lies with how the priorities of the nodes are determined. DGA randomly assigns priorities, assuming that the random numbers assigned are unique among the nodes (Algorithm 9, Line 5). DFA assigns priorities based on the orientation with the most number of targets covered (Algorithm 9, Lines 7-8). Since the priority values assigned by DFA are not necessarily unique, a mechanism for tie-breaking is needed (Algorithm 9, Lines 13-23). Algorithm 9 decides on the priority to assign using the binary variable DFAStage1 (Algorithm 9, Line 4), which is assumed to be a user input. A value of **0** for DFAStage1 denotes that the algorithm being represented is 2S-DGA, while a value of **1** denotes that the algorithm being represented is 2S-DFA. The stage the algorithm is in is denoted by the variable $\mathsf{CurrentState}_i$. The first stage of the algorithm can be found in Algorithm 9, Lines 11-26 while the second stage can be found in Algorithm 9, Lines 29-43.

In the first stage, a message received by the node is processed as long as the

message is from a node with a higher priority or while the supporting function `TimerFired` returns **FALSE** (Algorithm 9, Line 11). `TimerFired` is a function which returns **FALSE** when the timer has not fired yet, and **TRUE** after the timer has fired. We assume that the timer is a countdown timer which fires after a certain amount of time has elapsed after the timer is started. The timer is started (and the amount of time the timer will wait for specified) using the supporting function `StartTimer`. The timer is simultaneously stopped and reset using the supporting function `StopTimer`. The three supporting functions are tabulated and described in Table 5.3.

Function	Input parameters	Return value	Description
`StartTimer`	time before timer fires	none	starts the timer
`StopTimer`	none	none	stops and resets the timer
`TimerFired`	none	binary	returns **TRUE** if timer has fired; returns **FALSE** if otherwise

Table 5.3: Supporting functions for Algorithm 9 and Algorithm 10

The messages processed in the first stage cause the data structures seen_targets and unseen_targets to be updated (Algorithm 9, Line 24), and the function `TwoStageSetAndAnnounceBestOrientation` (Algorithm 10) to be called. Similar to its counterpart for TIA-DGA, `TwoStageSetAndAnnounceBestOrientation` chooses the orientation with the highest additional utility. Unlike Algorithm 8 however, Algorithm 10 defines utility differently. While Algorithm 8 defines utility as a weighted sum of the number of covered targets and the number of syndromes (Algorithm 8, Line 8), utility for Algorithm 10 in the first stage is solely determined by the number of covered targets (Algorithm 10, Line 9). If the chosen orientation of the node is changed, the choice is broadcast (Al-

gorithm 10, Line 21). At the end of Algorithm 10, the timer is started with the supporting function `StartTimer`, and the timer duration is specified to be equivalent to the value TIMEOUT. TIMEOUT is an amount of time sufficient for the entire network to be covered by DGA or DFA. The timer is stopped or reset every time a message is received (Algorithm 9, Line 12), assuming that the timer has not fired first. The situation of a message from a neighbouring node in Stage 1 arriving after the node's timer has fired or elapsed should *never* happen if the variable TIMEOUT is properly set.

The firing of the timer indicates that that the first stage is over and the algorithm now moves to the second stage (Algorithm 9, Line 27). It must be noted that nodes that got an orientation assignment from the first stage will no longer participate in the second stage (Algorithm 9, Line 28).

The operation of the second stage is different from that of the first stage primarily in that the timer no longer plays a role in the admission of packets or messages (Algorithm 9, Line 29) and Algorithm 10 now defines utility based on the number of syndromes generated (Algorithm 10, Line 11).

5.7 Methodology and simulation results

5.7.1 Methodology

We implement the algorithms and run the experiments in Matlab. While simulators such as TOSSIM [64] are able to represent and simulate WSNs (and WSN nodes) at a greater level of detail, in this work we assume a highly idealized communication channel (Section 5.1). As such, a simulation framework based on Matlab will suffice for our purposes. Different parameters are varied in different simulations; however, unless otherwise stated, default parameters for a setup are: 50 targets and 50 sensors randomly distributed over a 50 x 50 space, with each sensor having a sensing radius of 10, and 4 possible sensing orientations. Similar to Figure 5.1, the axes dividing the sensing regions are aligned to a North-East-West-South orientation; however, it should be noted that the

Algorithm 9 2-stage Target Identifiability-Aware Distributed Generic Algorithm

Inputs: α; $\phi_{i,j}$s; W; Q_i; Φ_i; Φ'_i; DFAStage1 - binary variable, **1** if desired first stage is DFA, **0** if DGA
Output: j_chosen - chosen orientation for the sensor node
Set $\mathsf{CurrentState_i} = \mathbf{1}$
if DFAStage1 == **0 then**
    Set priority $\mathsf{PR_i}$ = unique random number
else
    $\mathsf{j\_highest} \leftarrow \arg\max_{1 \le j \le W} |\phi_{i,j}|$
    Set priority $\mathsf{PR_i} = |\phi_{i,\mathsf{j\_highest}}|$
end if
`TwoStageSetAndAnnounceBestOrientation`();
while (`TimerFired`() == $FALSE$) AND (a protocol message is received from sensor $n \in Q_i$ s.t. $\mathsf{PR_n} \ge \mathsf{PR_i}$) **do**
    `TimerStop`();
    if $\mathsf{PR_n} == \mathsf{PR_i}$ **then**
        if $\Phi_i > \Phi_n$ **then**
            continue;
        else
            if $n > i$ **then**
                continue;
            else
                stop processing message, throw it away
            end if
        end if
    end if
    Based on the content of the message and Equation 5.7, update seen_targets matrix and unseen_targets matrix
    `TwoStageSetAndAnnounceBestOrientation`();
end while
Set $\mathsf{CurrentState_i} = \mathbf{2}$

results should not be any different if it is otherwise. Also, unless otherwise stated, results presented are the averages of 1000 experiment runs.

The effect of α on the heuristic algorithms' performance is evaluated in Section 5.7.2. The effect of the number of sensors, number of targets, etc., is evaluated in Section 5.7.3. Section 5.7.4 will compare the execution times of TIA-CGA and the two centralized 2-stage algorithms. A limited validation of the heuristic algorithms' performance against that optimal solutions is provided in Section 5.7.5.

```
28: if j_chosen == 0 then
29:     while a protocol message is received from sensor n ∈ Q_i s.t. PR_n ≥ PR_i do
30:         if PR_n == PR_i then
31:             if Φ_i > Φ_n then
32:                 continue;
33:             else
34:                 if n > i then
35:                     continue;
36:                 else
37:                     stop processing message, throw it away
38:                 end if
39:             end if
40:         end if
41:         Based on the content of the message and Equation 5.7, update seen_targets matrix and unseen_targets matrix
42:         TwoStageSetAndAnnounceBestOrientation();
43:     end while
44: end if
```

5.7.2 Effect of *alpha* on heuristic algorithms

We test the effect of the parameter α on the performance of TIA-CGA and TIA-DGA using a setup with 50 targets and 50 sensors. Figure 5.3 plots the % of targets covered, % of sensors that are active, and the number of syndromes. Five values for α are tested: 0, 0.25, 0.5, 0.75, and 1.0. To provide a point of comparison, we also plot the metrics for 2S-CGA, 2S-CFA, 2S-DGA, and 2S-DFA. As expected, the plots for the four are horizontal lines, since they are not parameterizable by α.

The first thing that must be noted from Figure 5.3 is how similar the results are for $\alpha = 0.25$, $\alpha = 0.5$, and $\alpha = 0.75$. This signifies that while the algorithms allow users to specify the acceptable level of trade-off between coverage and identifiability, in reality and practice, sensor-target setups offer a limited number of possible configurations and solutions. Therefore different values of α (except 0 and 1.0) can result in the same solution. This does not mean however that α does not matter, as the results for $\alpha = 0$ and $\alpha = 1.0$ will show.

When $\alpha = 0$, the algorithm degenerates into a syndrome-maximizing algorithm.

Algorithm 10 Algorithm for setting and announcing best orientation, 2-stage version (`TwoStageSetAndAnnounceBestOrientation`())

```
1: j_old ← j_chosen
2: for j s.t. 1 ≤ j ≤ W do
3:     P_j ← number of rows in seen_targets matrix with all 0s, s.t. target corresponding to the row ∈ φ_{i,j}
4:     R_j ← number of unique rows in seen_targets matrix that can also be found in unseen_targets matrix, s.t. target corresponding to the seen_targets matrix row ∈ φ_{i,j}
5:     if P_j != 0 then
6:         R_j ← R_j + 1
7:     end if
8:     if CurrentState_i = 1 then
9:         U_j ← P_j
10:    else
11:        U_j ← R_j
12:    end if
13: end for
14: if arg max_{1≤j≤W} U_j == 0 then
15:    j_chosen ← 0
16: else
17:    j_chosen ← arg max_{1≤j≤W} U_j
18: end if
19: if j_chosen != j_old then
20:    Set orientation to j_chosen
21:    Send a protocol message including priority and φ_{i,j_chosen}
22: end if
23: if CurrentState_i = 1 then
24:    StartTimer(TIMEOUT);
25: end if
```

For TIA-CGA, $\alpha = 0$ utilizes the same number of sensors as the three middle α values: 30% of the total (Figure 5.3b). With $\alpha = 0$, the TIA-CGA covers slightly fewer targets than the solution for all other α values (Figure 5.3a) - 77% compared to $\alpha = 0.25$'s 79%, for instance. Surprisingly, $\alpha = 0$ actually has a slightly lower average number of syndromes compared to the three middle α values (Figure 5.3c) - 22.07 compared to $\alpha = 0.25$'s 22.41.

For TIA-DGA, $\alpha = 0$ utilizes a slightly higher number of sensors than the three middle α values: 36% compared to 35% for $\alpha = 0.25$ (Figure 5.3b). With $\alpha = 0$, TIA-DGA covers less targets than the solution for all other α values (Figure 5.3a) - 71% compared to $\alpha = 0.25$'s 78%, for instance. Surprisingly, at $\alpha = 0$, TIA-DGA actually has a lower average number of syndromes compared

to the three middle α values (Figure 5.3c).

$\alpha = 1.0$ signifies that only the number of covered target matters, and the algorithm degenerates into a coverage-maximizing algorithm.

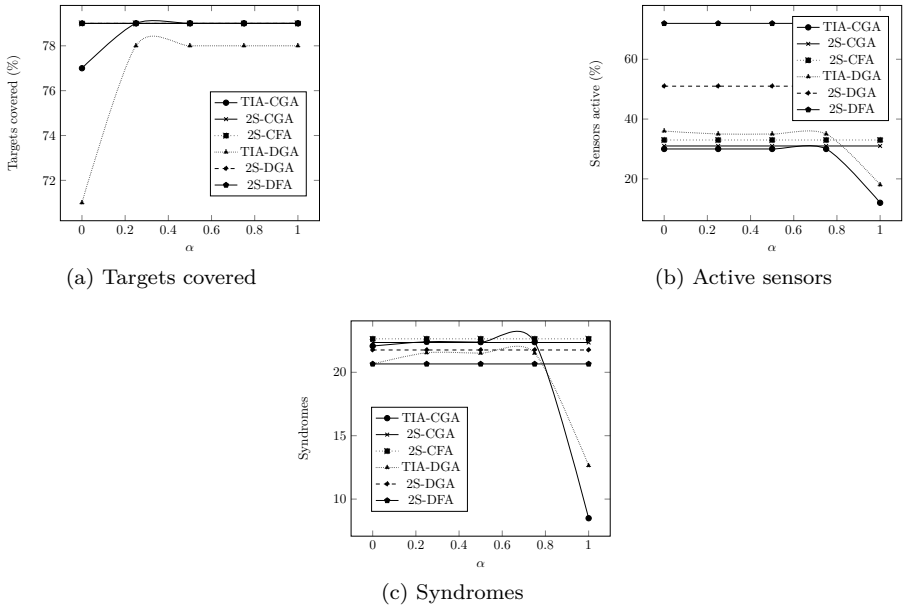
TIA-CGA at $\alpha = 1.0$ covers almost the same number of targets as the middle α values (Figure 5.3a), but does so with significantly less sensors: 12%, compared to 30% of $\alpha = 0.25$ (Figure 5.3b). With $\alpha = 1.0$, TIA-CGA ends up with a significantly smaller number of syndromes than at other α values (Figure 5.3c) - 8.49, against $\alpha = 0.25$'s 22.41.

TIA-DGA at $\alpha = 1.0$ also covers almost the same number of targets as the middle α values (Figure 5.3a), and also does so with significantly less sensors: 18%, compared to 35% of $\alpha = 0.25$ (Figure 5.3b). With $\alpha = 1.0$, TIA-DGA likewise ends up with a significantly smaller number of syndromes than at other α values (Figure 5.3c) - 12.64, against $\alpha = 0.25$'s 21.54.

In summary, Figure 5.3 illustrates that because of the limited number of solutions offered by sensor-target setups, the solutions for values of α between 0 and 1.0 do not differ by much from each other. However, a non-0, non-1.0 α value still offers a middle ground between $\alpha = 0$ and $\alpha = 1.0$; or more accurately, it offers the 'best of both worlds' in terms of targets covered and syndromes generated.

5.7.3 Main comparison

In this subsection, we test the effect of the number of sensors, number of targets, number of possible orientations, and the sensing radius on the metrics number of targets covered, number of sensors that are active, number of syndromes generated, and the system utility garnered. For the distributed algorithms we also include the *total* number of broadcasts made by *all* nodes. We assume that two nodes that can cover at least one target in common can communicate with one another (i.e., are one-hop neighbours in the network topology). We also assume that the nodes utilize a perfect media access control (MAC) protocol, and that there are no retransmissions due to interference (or hidden and exposed terminal



(a) Targets covered

(b) Active sensors

(c) Syndromes

Figure 5.3: Effect of α on heuristic algorithms. 50 targets, 50 sensors, 50x50 space, 4 sensing orientations, sensing radius = 10.

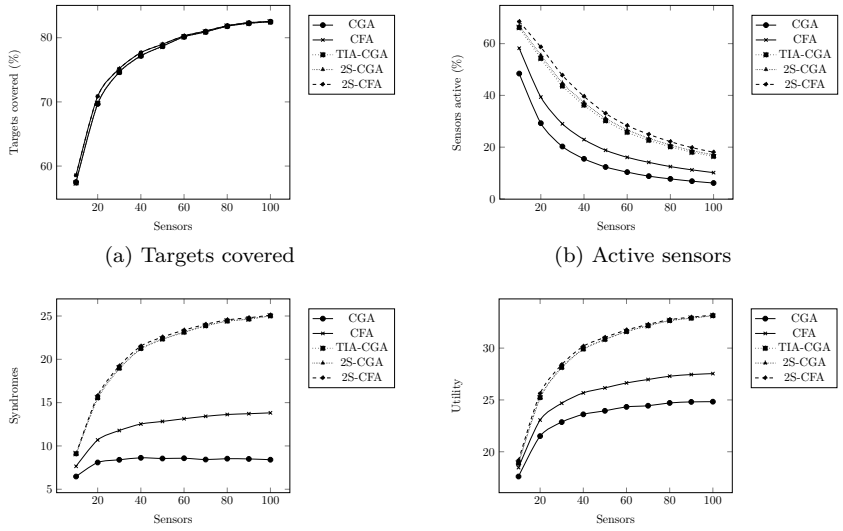
problems). Such an assumption is difficult to realise in actual implementations, but the results from the simulations will at least give us an approximate measure of the communication costs associated with the distributed heuristic algorithms.

Number of sensors

For the first simulation set, we vary the number of sensors from 10-100 and hold the number of targets constant at 50.

The results for the centralized algorithms are shown in Figure 5.4. As can be seen in Figure 5.4a, as more sensors are added, the % of covered targets increases, but the increase eventually slows down. All three heuristic algorithms track CGA and CFA very closely. The diminishing increase in the number of covered targets indicates that the system is becoming more and more over-provisioned, with an increasing number of sensors becoming idle since they no longer have any targets to cover or syndromes to contribute. This is consistent with Figure 5.4b which shows the drop in % of sensors which are active. Among the heuristic algorithms, 2S-CFA consistently utilizes the most sensors, followed

by 2S-CGA, and then TIA-CGA. In Figure 5.4c, we see that consistent with their aims, all three heuristic algorithms consistently have significantly higher number of syndromes than either CGA or CFA. The number of syndromes increases with the number of sensors, but the increase becomes more and more attenuated as the number of sensors increases. The plateauing in the number of syndromes occurs much earlier for CGA and CFA. The same pattern is seen in Figure 5.4d, which shows the system utility. 2S-CFA has a slight but consistent advantage over the two other heuristic algorithms when it comes to the system utility garnered.



(a) Targets covered (b) Active sensors

(c) Syndromes (d) Utility

Figure 5.4: Effect of varying the number of sensors on *centralized* algorithms. α = 0.5, 50 targets, 50x50 space, 4 sensing orientations, sensing radius = 10.

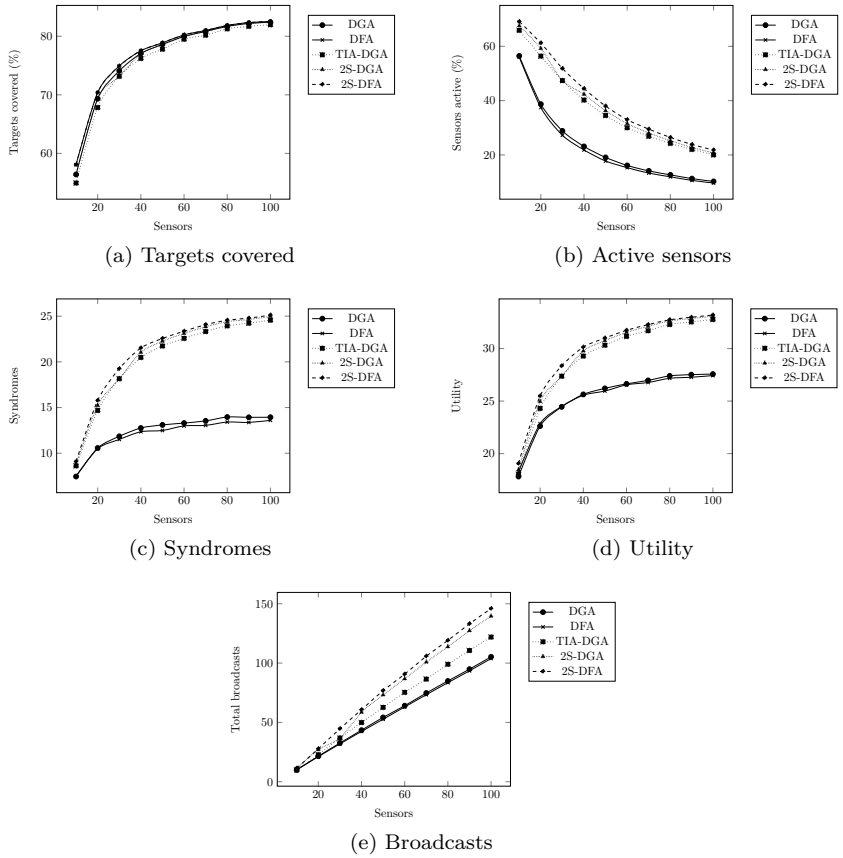
The results for the distributed algorithms are shown in Figure 5.5. As can be seen in Figure 5.5a, as more sensors are added, the % of covered targets goes up, but the increase eventually slows down. The % of covered targets are very similar for all algorithms, with TIA-DGA lagging just very slightly behind the others. The plateauing indicates that the system is becoming more and more over-provisioned, with more and more sensors becoming idle since they no longer have any targets to cover or syndromes to contribute. Consistent

with this, Figure 5.5b shows the drop in % of active sensors. The plots are quite similar for DGA and DFA, with the plots of the three heuristic algorithms being higher - this is because TIA-DGA, 2S-DGA, and 2S-DFA utilize the additional sensors for increasing the number of syndromes. Between the three heuristic algorithms, TIA-DGA utilizes a slightly lower number of active sensors than the two 2-stage algorithms. In Figure 5.5c, we see that consistent with their aims, TIA-DGA, 2S-DGA, and 2S-DFA consistently have a higher number of syndromes than either DGA or DFA. The number of syndromes increases with the number of sensors, but the increase becomes more and more attenuated as the number of sensors increases. The number of syndromes plateau much earlier for DGA and DFA. Between the three distributed heuristic algorithms, and with respect to the number syndromes, 2S-DFA performs best, followed very closely by 2S-DGA, and then TIA-DGA. The pattern for the number of syndromes is repeated for the system utility (Figure 5.5d). As for the number of broadcasts made (Figure 5.5e), all five algorithms follow a linear (increasing) relationship with the number of sensors. The number of broadcasts that DGA and DFA made are highly similar. The slope for the TIA-DGA is slightly higher than that of DGA and DFA. Both 2S-DGA and 2S-DFA have noticeably higher slopes than TIA-DGA, with 2S-DFA having a slightly higher slope than 2S-DGA.

Number of targets

For the second simulation set, we vary the number of targets from 10-100 and hold the number of sensors constant at 50.

The results for the centralized algorithms are plotted in Figure 5.6. Figure 5.6a shows that the % of targets covered remains constant all throughout the values tested (this means that the absolute number of targets covered *increases*). As the number of targets increases, the % of sensors that are active also increases (Figure 5.6b). We also see in Figure 5.6b that compared to CGA and CFA, the three heuristic algorithms consistently have a higher number of sensors that are active. Among the heuristic algorithms, 2S-CFA consistently utilizes more



(a) Targets covered

(b) Active sensors

(c) Syndromes

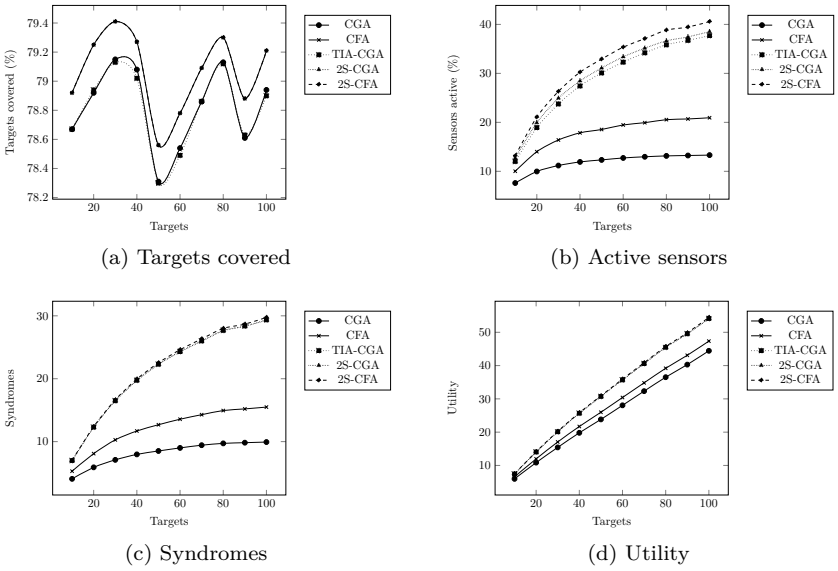
(d) Utility

(e) Broadcasts

Figure 5.5: Effect of varying the number of sensors on *distributed* algorithms. α = 0.5, 50 targets, 50x50 space, 4 sensing orientations, sensing radius = 10.

sensors than 2S-CGA, which in turn, consistently utilizes more than TIA-CGA. Consistent with their aims, Figure 5.6c shows that the heuristic algorithms consistently have more syndromes than CGA and CFA. The same can also be said about the system utility, shown in Figure 5.6d. When it comes to the system utility garnered, 2S-CFA once again has a slight but consistent advantage over the two other heuristic algorithms.

The results for the distributed algorithms are plotted in Figure 5.7. Figure 5.7a shows that the % of targets covered remains constant all throughout the values tested, with only a very slight variation across the values tested (this means that the absolute number of targets covered *increases*). It can also be seen in the plot that TIA-DGA trails the other four algorithms when it comes to the % of targets



(a) Targets covered

(b) Active sensors

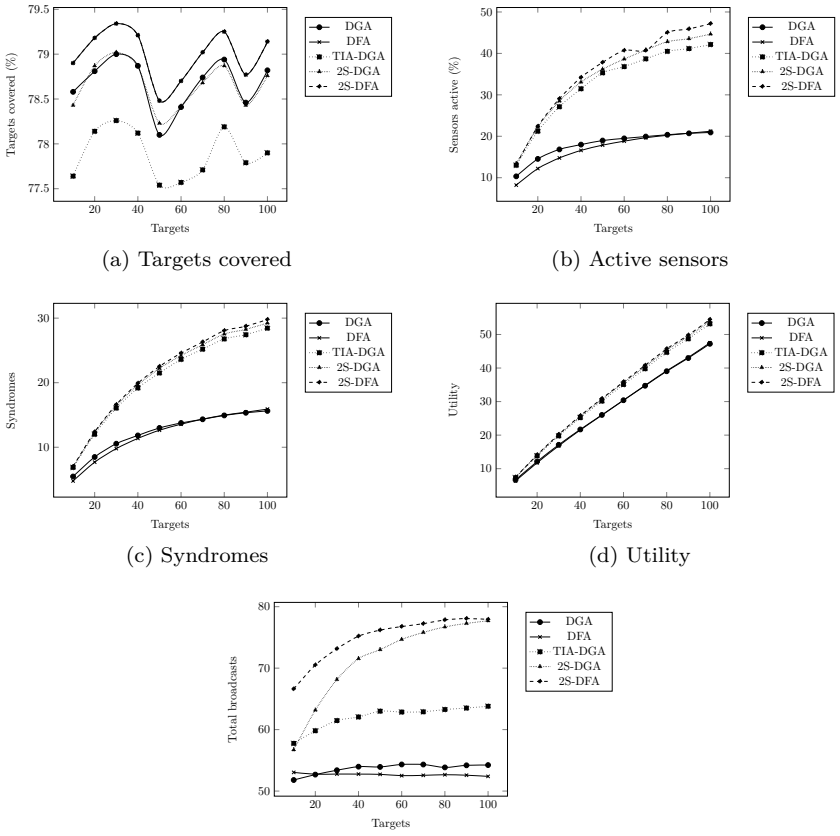
(c) Syndromes

(d) Utility

Figure 5.6: Effect of varying the number of targets on *centralized* algorithms. α = 0.5, 50x50 space, 50 sensors, 4 sensing orientations, sensing radius = 10.

covered. The number of active sensors increases for all algorithms, although the increase is much steeper for the TIA-DGA, 2S-DGA and 2S-DFA than DGA or DFA (Figure 5.7b). The increase in the number of active sensors can also be seen to plateau, as already active sensors prove sufficient to cover the targets that are added. Between the three distributed algorithms that are target-identifiability aware, the increase is greatest for 2S-DFA, followed by 2S-DGA, and finally, TIA-DGA. Consistent with the algorithms' aims, Figures 5.7c and 5.7d show that TIA-DGA, 2S-DGA and 2S-DFA have consistently higher number of syndromes and system utility than both DGA and DFA, with the difference increasing with the number of targets. Between the three distributed algorithms for MTIAUMS, the same ordering seen before in Figure 5.7b is again repeated in Figures 5.7c and 5.7d: 2S-DFA closely followed by 2S-DGA, which is then closely followed by TIA-DGA. As for broadcasts, it can be seen in Figure 5.7e that DGA and DFA utilize significantly less broadcast messages than the other three algorithms, and the number of broadcast messages stay more or less the same even when the number of targets is increased. In comparison, the num-

ber of broadcasts made by TIA-DGA, 2S-DGA and 2S-DFA increases with the number of targets. 2S-DFA consistently makes more broadcasts than TIA-DGA. 2S-DGA starts with less broadcasts than TIA-DGA, but eventually catches up with 2S-DFA.



(a) Targets covered

(b) Active sensors

(c) Syndromes

(d) Utility

(e) Broadcasts

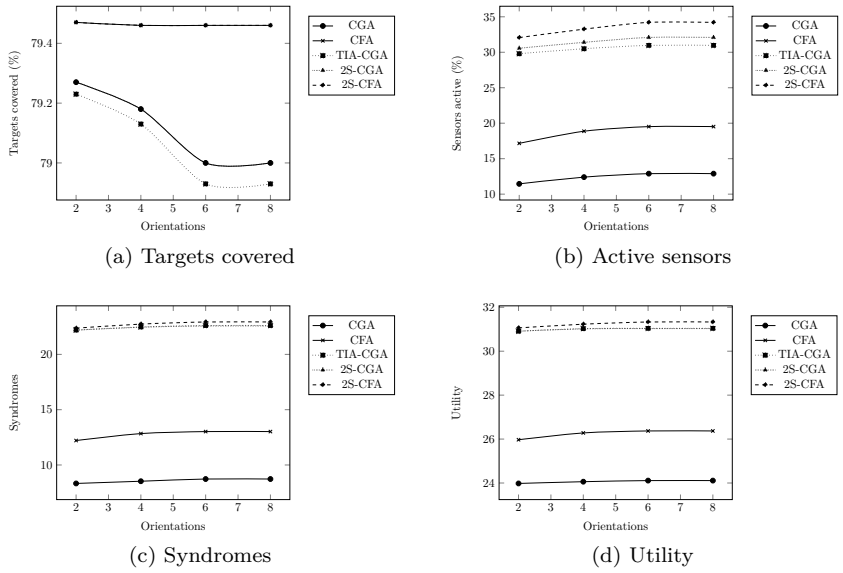
Figure 5.7: Effect of varying the number of targets on *distributed* algorithms. α = 0.5, 50x50 space, 50 sensors, 4 sensing orientations, sensing radius = 10.

Number of sensor orientations

For the third simulation set, we vary the number of possible sensor orientations from 2-8 and hold the number of sensors and targets constant at 50.

The results for the centralized algorithms are plotted in Figure 5.8. It must be noted that an increase in the number of possible sensor orientations implies a decrease in the size of the sensed region. In Figure 5.8a it can be seen that when

it comes to the number of targets covered, the increase in the number of possible sensor orientations only slightly affects CFA and 2S-CFA. It has the effect of decreasing the number of covered targets for TIA-CGA, CGA, and 2S-CGA, although the decrease seems to plateau after 6. As for the number of active sensors (Figure 5.8b), the increase in the number of possible sensor orientations results in the increase in the number of active sensors for all algorithms. The number of syndromes for the three heuristic algorithms slightly increases as the number of possible sensor orientations increases (Figure 5.8c). The slight increase is brought about by the increase in the number of intersecting sensed regions, which is a consequence of the increased number of active sensors. The pattern seen in the number of syndromes is carried over to the system utility garnered by the algorithms (Figure 5.8d).



(a) Targets covered

(b) Active sensors

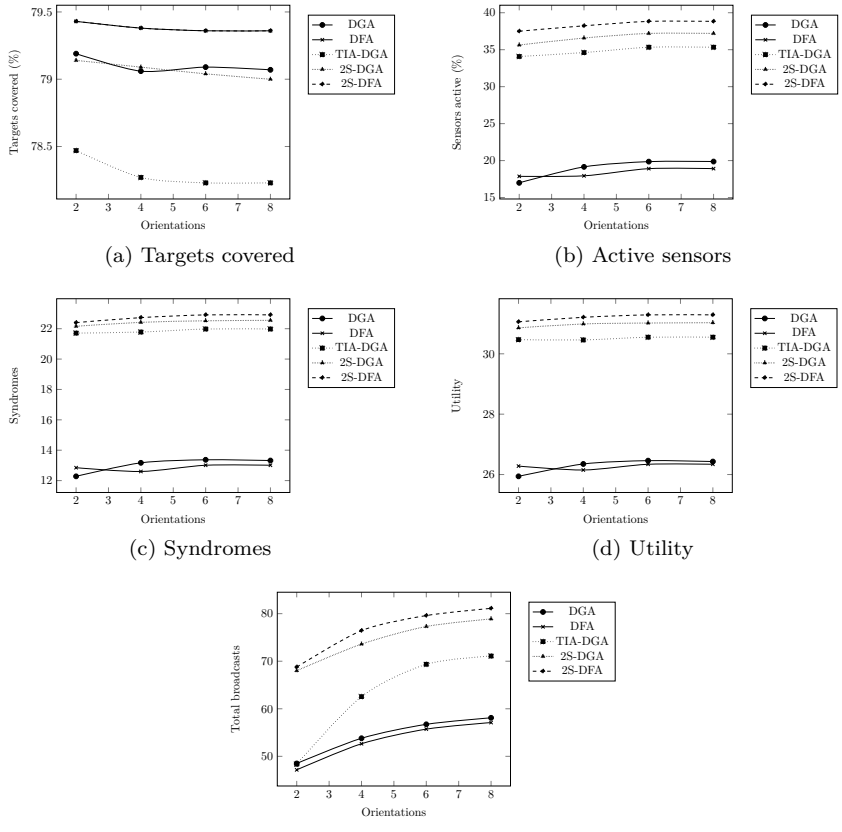
(c) Syndromes

(d) Utility

Figure 5.8: Effect of varying the number of possible sensor orientations on *centralized* algorithms. $\alpha = 0.5$, 50x50 space, 50 sensors, 50 targets, sensing radius = 10.

The results for the distributed algorithms are plotted in Figure 5.9. In Figure 5.9a it can be seen that the increase in the number of possible sensor orientations causes a slight decrease in the number of targets covered for all algorithms. The increase in the number of possible sensor orientations also has the

effect of slightly increasing the number of active sensors (Figure 5.9b), which is to be expected as the area covered by each sensor decreases. The increase in the number of active sensors also causes an increase in intersection of active sensed regions, resulting in an increase in the number of syndromes (Figure 5.9c). Similar to the centralized algorithms, even with the slight decrease in the number of covered targets, the increase in the number of syndromes more than compensates and the system utility garnered increases for all algorithms (Figure 5.9d). As for the number of broadcasts, the increase in the number of possible sensor orientations causes an increase in the number of broadcasts for all algorithms (partially to be expected due to the increase in the number of active sensors), although the increase is especially pronounced for TIA-DGA (Figure 5.9e).



(a) Targets covered

(b) Active sensors

(c) Syndromes

(d) Utility

(e) Broadcasts

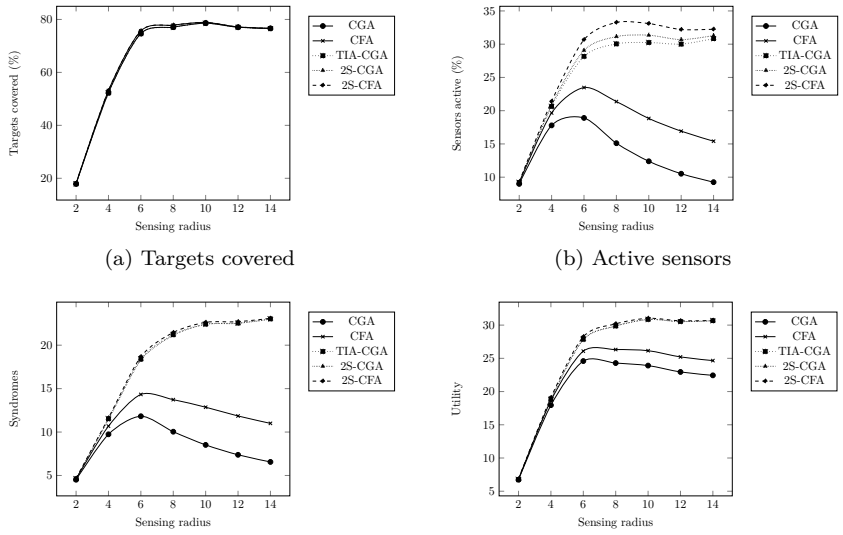
Figure 5.9: Effect of varying the number of possible sensor orientations on *distributed* algorithms. $\alpha = 0.5$, 50x50 space, 50 sensors, 50 targets, sensing radius = 10.

Sensing radius

For the fourth simulation set, we vary the sensing radius from 2-8 and hold the number of sensors and targets constant at 50.

The results for centralized algorithms are plotted in Figure 5.10. In Figure 5.10a it can be seen that as the sensing radius increases, more targets are covered as sensors are able to reach previously unreachable targets. The increase however eventually flattens out. The trend holds true for all the algorithms. To cover the targets that can now be seen by the sensors, more sensors are activated (Figure 5.10b). Nevertheless, the increase in the number of active sensors eventually plateaus for TIA-CGA, 2S-CGA and 2S-CFA. In contrast, the number of active sensors actually *decreases* for CGA and CFA after some point - as the sensing radius of sensors grows, fewer and fewer sensors are needed to cover the targets (and coverage is solely the concern of CGA and CFA). The decrease is not seen in the TIA-CGA, 2S-CGA and 2S-CFA (at least not yet in the values tested) because the multiple coverage is used to increase the number of syndromes (Figure 5.10c). The increase in the number of covered targets (at least in the first few values) and the increase in the number of syndromes lead to increased system utility for all algorithms, although that for CGA and CFA eventually decreases (Figure 5.10d). The decrease in system utility for CGA and CFA is due to the decrease in the number of syndromes, which, in turn, is caused by the decrease in the number of covering sensors.

The results for distributed algorithms are plotted in Figure 5.11. The patterns seen in Figures 5.10a-5.10d can also be seen in their counterparts in Figures 5.11a-5.11d. The increase in the number of targets that can be covered (Figure 5.11a), as well as the increase in the number of active sensors (Figure 5.11b), lead to the increase in the number of broadcasts (Figures 5.11e) - however, the increase for TIA-DGA is markedly less than that of 2S-DGA and 2S-DFA. The increases for DGA and DFA are even less than that of TIA-DGA, and also flatten out very early.



(a) Targets covered

(b) Active sensors

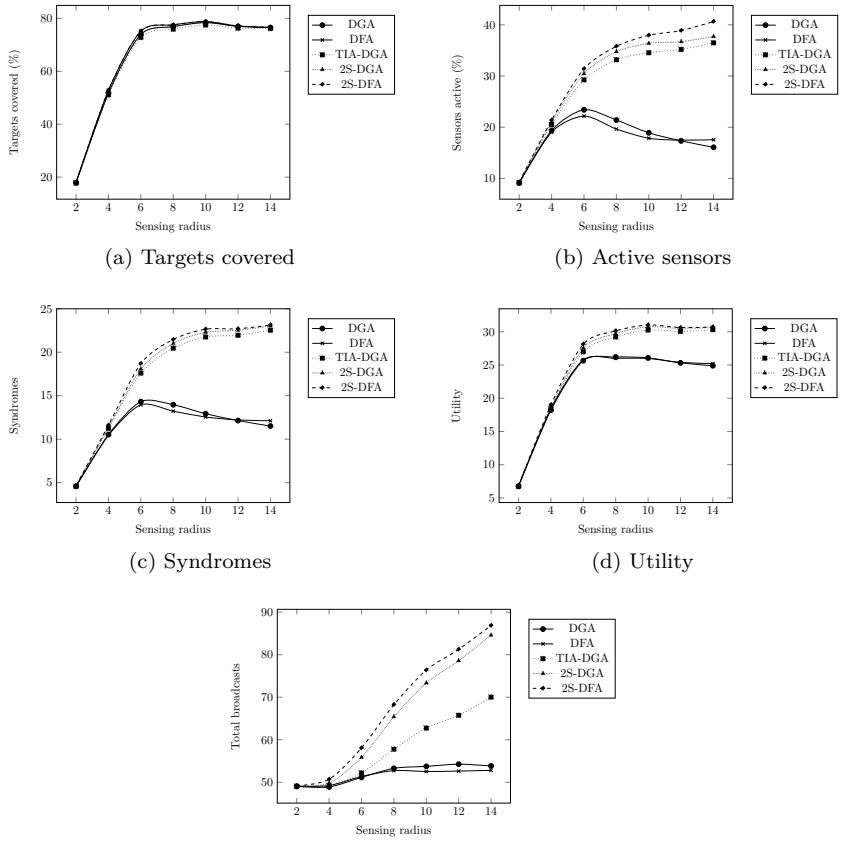
(c) Syndromes

(d) Utility

Figure 5.10: Effect of varying the sensing radius on *centralized* algorithms. $\alpha = 0.5$, 50x50 space, 50 sensors, 50 targets, 4 sensing orientations.

5.7.4 Comparison of execution times of centralized heuristic algorithms

The primary advantage of (and rationale for) the centralized 2-stage algorithms is their lower time complexity compared to that of TIA-CGA. To validate this, we plot the average algorithm execution times for the centralized heuristic algorithms (as when they are used in Section 5.7.3 and Section 5.7.3) in Figure 5.12. Figure 5.12a shows the execution times for the simulation set where the number of sensors is varied, while Figure 5.12b shows the execution times for the simulation set where the number of targets is varied. Comparing Figure 5.12a and Figure 5.12b, it becomes apparent that the number of sensors has a greater effect on the growth of the execution time than the number of targets. It can also be seen in both plots that TIA-CGA consistently has longer execution times than the two 2-stage algorithms.



(a) Targets covered

(b) Active sensors

(c) Syndromes

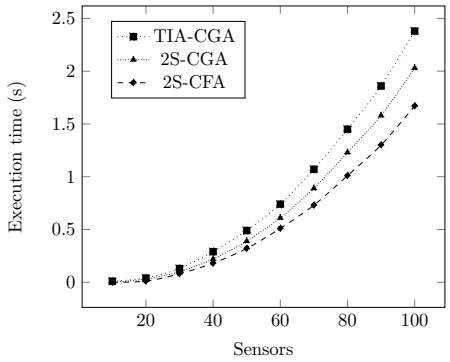
(d) Utility

(e) Broadcasts

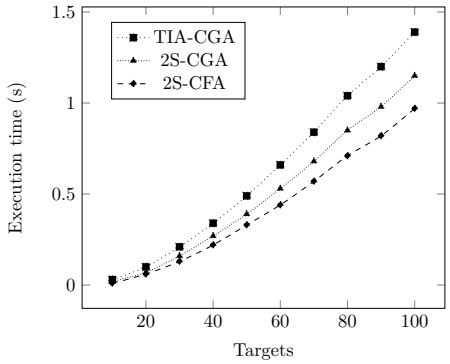
Figure 5.11: Effect of varying the sensing radius on *distributed* algorithms. $\alpha = 0.5$, 50x50 space, 50 sensors, 50 targets, 4 sensing orientations.

5.7.5 Limited comparison of heuristic solutions to optimal solutions

To provide validation (even if just a limited one) of how close the heuristic solutions are to the optimal solutions, we solve a limited number of problems (20 for each setup) using brute force. The number of targets is varied from 50 to 100, while the number of sensors is set constant to 10. The targets and sensors are distributed over a 30 x 30 space, with α set to 0.5, the number of sensing regions to 4, and the sending radius to 5. The results of the simulations are plotted in Figure 5.13 (centralized) and Figure 5.14 (distributed). The values plotted in Figure 5.13 and Figure 5.14 are the averages of 20 runs.



(a) Number of sensors varied

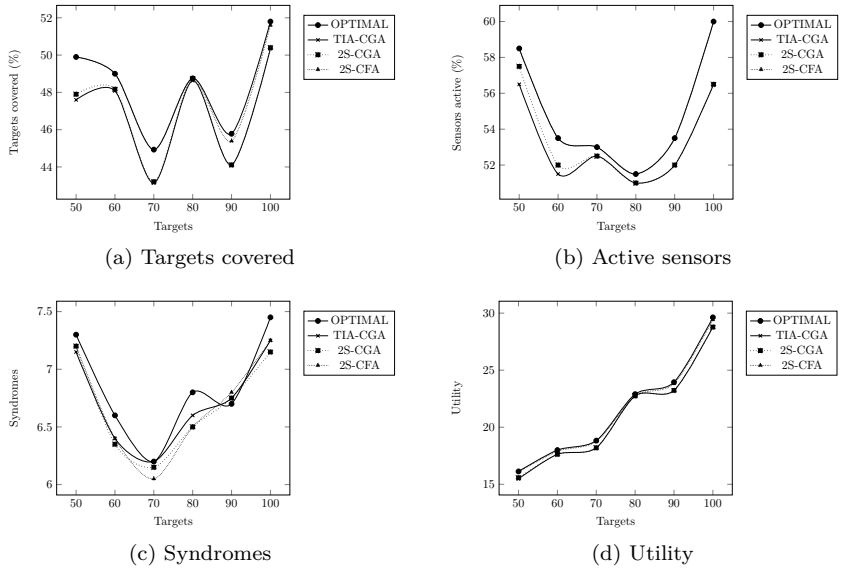


(b) Number of targets varied

Figure 5.12: Average execution times for the heuristic algorithms. $\alpha = 0.5$, 50x50 space, 4 sensing orientations, sensing radius = 10.

It can be seen in Figure 5.13 and Figure 5.14 that the heuristic solutions approximate the optimal solutions well. The system utility, which is the value being maximized, has two components: the number of targets covered and the number of syndromes generated. Figure 5.13 and Figure 5.14 show that it is possible for a heuristic algorithm to produce solutions that have more than the optimal solution in a certain component: for example, for targets = 90, all centralized heuristic solutions (Figure 5.13c) and 2S-DFA (Figure 5.14c) actually have more syndromes than the optimal solution. For all the setups tested, the best performance (in terms of system utility) by a centralized heuristic algorithm (Figure 5.13d) is by 2S-CFA (targets = 50), which comes within 0.3% of the optimal solution's system utility. The worst performance by a centralized algorithm is by TIA-CGA (targets = 50), which differs from the optimal solution

by 4.0%. For the distributed algorithms (Figure 5.14d), the best performance is by 2S-DFA (targets = 80), with only 0.52% difference, while the worst is by TIA-DGA, with 8.25% difference.



(a) Targets covered

(b) Active sensors

(c) Syndromes

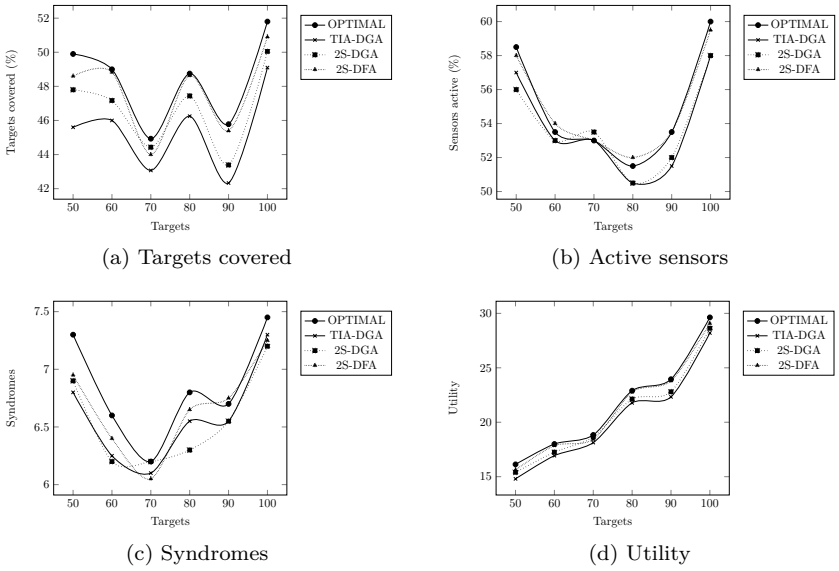
(d) Utility

Figure 5.13: Validation of solutions produced by *centralized* heuristic algorithms against optimal solutions. $\alpha = 0.5$, 10 sensors, 30x30 space, 4 sensing orientations, sensing radius = 5.

5.8 Related work

The problem of maximizing target-based coverage for directional sensors is first formalized in [2] as the Maximum Coverage Minimum Sensors (*MCMS*) Problem. The MCMS problem's Integer Linear Programming (ILP) formulation is also given in [2], along with 2 heuristic-based solutions: a centralized algorithm (Centralized Greedy Algorithm, or *CGA*) and a distributed algorithm (Distributed Greedy Algorithm, or *DGA*).

Munishwar and Ab-Ghazaleh [114] also deals with the problem of target coverage maximization, but in the specific context of visual sensor networks (cameras). The main difference between visual sensors and general sensors is the concept of R_{Min} or the *minimum* distance required between the sensor and



(a) Targets covered (b) Active sensors

(c) Syndromes (d) Utility

Figure 5.14: Validation of solutions produced by *distributed* heuristic algorithms against optimal solutions. $\alpha = 0.5$, 10 sensors, 30x30 space, 4 sensing orientations, sensing radius = 5.

the target for the latter to see the former. Munishwar and Ab-Ghazaleh [114] proposes alternatives to [2]'s CGA and DGA, called *CFA* (Centralized Force-based Algorithm) and *DFA* (Distributed Force-based Algorithm), respectively.

Another common problem in directional sensor networks is finding and scheduling cover sets. Cover sets are sets of sensors (and their orientations) that ensure that all targets are covered. It must be noted that finding cover sets is equivalent to finding network configurations repeatedly, each time removing the cover set-comprising nodes from the set of viable nodes. The need for cover sets is primarily motivated by the limited lifetimes of nodes. By finding several such cover sets and scheduling their activation, the targets will be covered for a longer period of time than if only a single cover set is found. Notable studies on cover sets are [17] and [161].

This study is not the first study to go beyond maximization of the number of covered targets.

Fusco and Gupta [39] deals with the problem of providing k-coverage to a given set of targets (or area) using a minimum number of sensors - a problem

they call *SODkC*, or Selecting and Orienting D-sensors for k-Coverage. The problem is shown to be NP-hard, and they propose two greedy solutions to the problem, a centralized algorithm and a distributed algorithm. Wang et al [159], on the other hand, deals with the case wherein the targets have different coverage requirements.

It must be noted however, that the problem of maximizing coverage or meeting coverage requirements is different than that of maximizing target identifiability. For instance, if what is wanted is to satisfy a certain k-coverage requirement, when two sensors cover *exactly* the same set of targets, both should be activated since by doing so the coverage of each target covered increases. However, if what is wanted is the maximization of target identifiability, only one of the sensors should be activated, as activating both adds no new syndrome to the system.

5.9 Summary

It is important to recognize that the assumption of built-in target identifiability needs to be re-examined in directional sensor networks, as it does not always hold true. To this end, we introduce the concept of *syndromes* which facilitates the measurement of target identifiability in a directional sensor network - this leads to the definition of the Maximum Target Identifiability-Aware Utility with Minimum Sensors (MTIAUMS) problem, which we prove to be NP-hard. We also introduce heuristic algorithms for orienting sensors in a directional sensor network. The algorithms take into account not just the number of targets covered but also their identifiability when finding network configurations.

We introduce three centralized heuristic algorithms: Target Identifiability-Aware Centralized Greedy Algorithm (TIA-CGA), 2-stage Target Identifiability-Aware Centralized Greedy Algorithm (2S-CGA), and 2-stage Target Identifiability-Aware Centralized Force-based Algorithm (2S-CFA). TIA-CGA is parameterizable with α which lets users specify the desired level of trade-off between coverage and identifiability. Simulations show that α loses sensitivity or differ-

entiability in the middle range of values because of the limited number of configurations possible for any sensor-target setup; nevertheless, the middle values still provide a middle ground, solutions-wise, between those given by extreme α values. The 2-stage algorithms on the other hand are not parameterizable, instead covering the targets first (using algorithms designed for MCMS) and then using the remaining sensors to increase the identifiability of covered targets. The 2-stage algorithms also tend to have shorter runtimes than TIA-CGA. Of the three heuristic algorithms, 2S-CFA in most cases perform best in the simulations carried out, followed by 2S-CGA, and then TIA-CGA. 2S-CFA however, also has the tendency to use more nodes than 2S-CGA or TIA-CGA.

We introduce three distributed heuristic algorithms: Target Identifiability-Aware Distributed Greedy Algorithm (TIA-DGA), 2-stage Target Identifiability-Aware Distributed Greedy Algorithm (2S-DGA), and 2-stage Target Identifiability-Aware Distributed Force-based Algorithm (2S-DFA). Similar to TIA-CGA, TIA-DGA is parameterizable with α. 2S-DGA and 2S-DFA are the distributed counterparts of 2S-CGA and 2S-CFA, respectively. In most of our simulations, 2S-DFA produces more syndromes and slightly higher system utilities than 2S-DGA and TIA-DGA.

While the causes of performance difference between heuristic algorithms are difficult to justify, we suspect that the slight superiority of CFA-based algorithms from CGA-based ones stem from CFA's documented performance advantage over CGA in covering targets[114]. We suspect that better performance at the first stage (target covering stage) leads to better options (and performance) in the second stage (syndrome increasing stage).

CHAPTER 6

Conclusion

In this thesis we enumerate four key areas that researchers can focus on to hasten the process of making energy-harvesting noise-sensing WSNs a practical reality. For each area, we provide a specific example and contribution, presented in Chapters 2-5.

The first key area is that of *new and emerging energy storage technologies*, and how current algorithms and infrastructures must be modified to take advantage of them. In Chapter 2, we propose methods that mitigate the negative effects on performance of using thin-film solid state batteries. Thin-film solid state batteries have several advantages over supercapacitors and Li-ion batteries, but place limits of the duty cycle that can be adapted by the system. The methods we present in Chapter 2 enable the system to adapt to such limits and minimize their effects on system performance.

The second key area is the area of *currently accepted technical requirements*, and an assessment on whether they will indeed lead to the attainment of long-term goals. In Chapter 3, we review the capabilities and limitations of energy-harvesting noise-sensing WSNs. We show that current low-power energy-harvesting noise-sensing WSNs will not meet the specifications of existing noise codes, especially regarding the noise metric. We propose an alternative energy-harvesting noise-sensing WSN node design that does *not* conform with the requirements of existing noise codes, but is better-suited to being powered by energy harvesting.

The third key area is the area of *methodologies, specifically methodologies involved in testing designs.* In Chapter 4, we present a test methodology (and its associated test apparatus) that enables repeatable experiments and tests

for solar-powered WSNs. We present two versions of the test apparatus: a *centralized* version which enables the testing of wireless sensor *nodes*, and a *distributed* version which enables the testing of wireless sensor *networks*. We demonstrate through a series of experiments how the test methodology and the test apparatus can be used in empirically deriving hardware and software design parameters.

The final key area is the area of *future capabilities and functionalities*, and the techniques or algorithms that will enable users to take advantage of them. In Chapter 5, we present heuristic algorithms that enable directional sensor networks to identify targets or event sources. Such a capability is relevant to noise-sensing WSNs, since noise-sensing WSNs can be readily turned into directional sensor networks by having the nodes use directional (instead of omnidirectional) microphones.

It must be noted that most of the contributions that we made in this thesis however, are not limited in scope to energy-harvesting noise-sensing WSNs.

The methods for optimizing charge times, presented in Chapter 2, can be used in *any* energy-harvesting WSN node that utilizes energy storage devices with high internal impedances, not just those that sense noise.

The methodology and test apparatuses we presented in Chapter 4 can be used with any solar-powered WSN node, and even networks of such nodes (through the distributed version of the test apparatus). The benefits of the methodology and test device actually go deeper, as the DUTs need not be WSN nodes - any embedded system powered by solar energy can also be tested.

The heuristic algorithms presented in Chapter 5 will benefit any directional sensor network in which target identifiability is important. The network need not be powered by energy harvesting or equipped with sensors for noise-sensing.

It must be emphasized that the examples given for each of the four areas are not exhaustive. Other aspects of each of the four areas can doubtless still be explored. We give examples of these possible future work in Chapter 1, where each of the four areas is first described in detail, as well as in each chapter that

describes our contributions.

6.1 Context, real-world results and further research

This work can trace its beginnings to the University of Warwick's involvement with the Center for Urban Science and Progress (CUSP) [76]. CUSP is a research centre based in New York which aims to utilize informatics to make cities productive, livable, equitable and resilient. CUSP has several partners both in industry and academe (of which the University of Warwick is one). The measurement of noise in urban areas is one of the original focus areas of CUSP, and in line with this, this project originally aimed to create a system for measuring noise while being powered by energy harvesting. Such a system will compliment the efforts of other CUSP activities, which include systems that identify noise sources (mainly through cellular phones or cellular phone-like devices tethered and powered from lamp posts) and efforts to crowdsource noise measurements (through an application that users will voluntarily run on their cellular phones). We aimed for system that can be deployed almost anywhere, can be easily maintained and with output that is always properly contextualized.

The results presented in Chapter 3 are mainly from our evaluation of the requirements that must be met by WSNs for them to be used in enforcing noise codes.

Together with Molecular Solar, a University of Warwick-based startup specializing in OPV cells, we were able to obtain support from the UK Technology Strategy Board, now called Innovate UK [85] (Project 131187/26835-183208, *OPV-based Energy Harvesting for Urban Noise Pollution*). The TSB project aimed to demonstrate that noise-measurement WSNs are a possible commercial application for OPVs, which is an emerging technology in the field of photovoltaics (Chapter 1.2.2). Due to unforeseen circumstances, Molecular Solar was later replaced by PolySolar [100], a University of Cambridge-based startup spe-

cializing in thin-film cells, as our industry partner in the project.

Our initial plan was to create a completely 'green' solution to the problem, leading to our involvement with novel energy storage devices. Our attempt to use solid state batteries, specifically the Enerchip, led to the algorithms presented in Chapter 2.

Also part of our original plan was the development of power management algorithms that take into account the short lifetime of OPV cells, specifically their health. The lack of a consistent and accurate methodologies for testing solar-powered wireless sensor nodes and networks led us to develop our own solution, which was presented in Chapter 4.

Eventually, delays in manufacturing, problems with the encapsulation process, and the lack of experimental data on the degradation of OPV cells (which would have allowed us to model the process and run simulations) led us to postpone efforts in creating power management algorithms that take into account the current health of OPV cells.

This research has culminated in a small deployment of wireless sensor nodes that measure noise while being powered by OPV-based solar panels (see Figures 6.1-6.3). Prioritizing simplicity over elegance, we decided to use more 'traditional' energy storage devices for our deployed system. In lieu of an OPV cell health-aware power management algorithm, we used a simpler power management algorithm based on Additive Increase Multiplicative Decrease (modeled after TCP's congestion control mechanism). The initial parameters for the power management algorithm were derived using the device presented in Chapter 4.

With the help of Sentec [132], an engineering design consulting firm specializing in smart metering technologies, we were able to produce a manufacturing-ready version of our design, named *Solar Owl* [133] (see Figure 6.4).

Admittedly, we were not able to achieve some of our more ambitious plans (a fully 'green' solution, OPV-specific power management algorithms). The experience however, has been highly instructive for us (and others). Our experience highlights the difficulties of working with and combining different emerg-

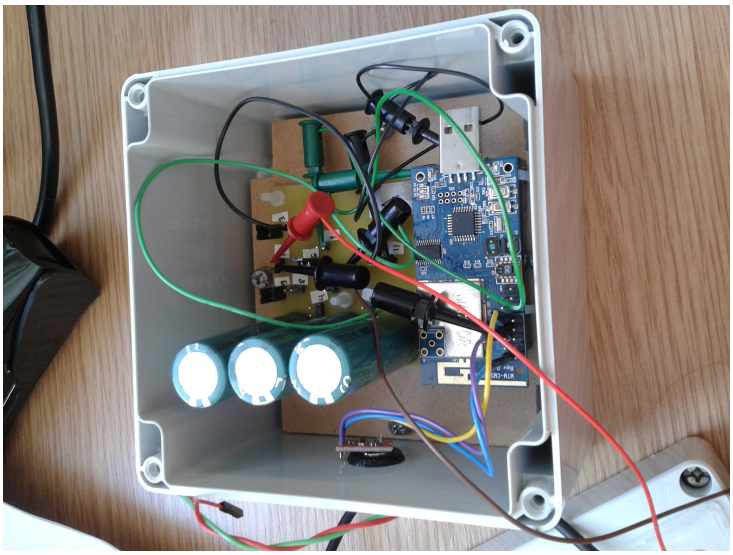


Figure 6.1: Device, internal view.

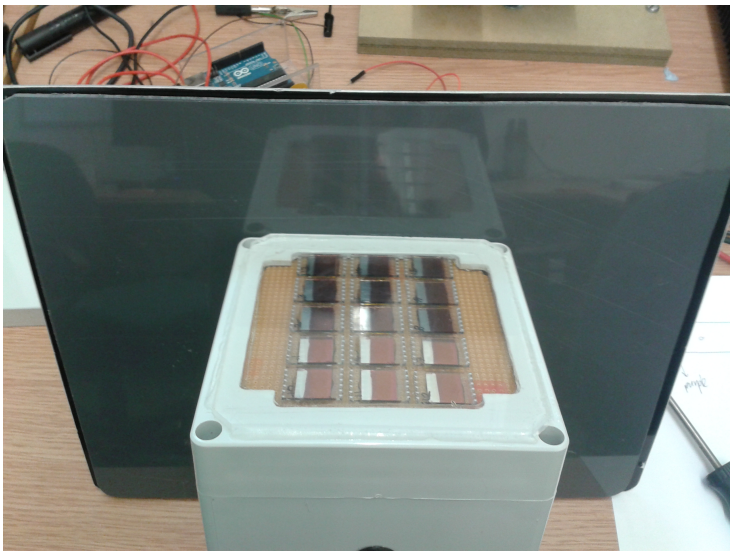


Figure 6.2: Assembled device.

ing technologies. For instance, while our algorithms enable the use of solid-state batteries for WSNs, it was still a challenge making them work with OPV cells, given the OPV cells' low output. As for power management algorithms that are OPV-specific, we believe that it is still a very promising research area. Such algorithms hold the key to OPV cells being truly practical to use with WSNs,

Figure 6.3: Assembled device in deployment.

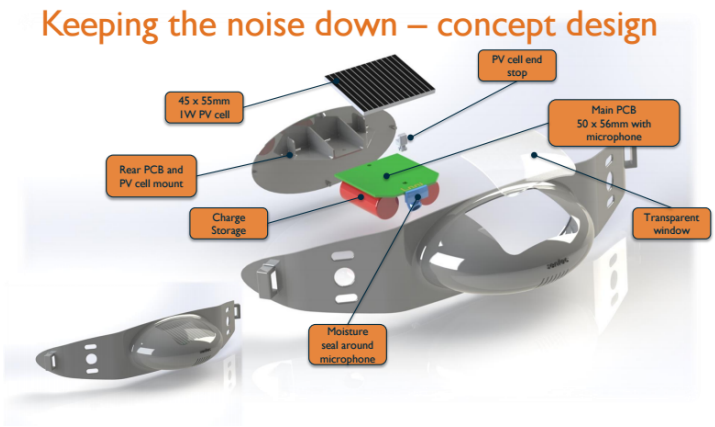


Figure 6.4: Solar Owl concept drawing.

unless OPV cell lifetimes improve to the point that they are comparable to those of crystalline silicon or thin film cells. Before such a study can be carried out however, the process of producing OPV cells reliably must first be perfected - or at the very least, extensive data on how they degenerate over time must first be collected.

A recurring theme in our series of studies is the importance of insights from allied or related disciplines, specifically from fields other than computer science

or engineering. The alternative node design presented in Chapter 3 has its beginnings from insights that came from psychologists and acoustic engineers (contact with whom was brought about by our involvement with the NYU-CUSP Project). The key concepts that make the methodology presented in Chapter 4 possible came from photovoltaic engineering and solar cell design, made accessible to us through collaboration with other research groups (academic and industrial) in the Universities of Warwick and Cambridge (brought about by our involvement in the TSB/Innovate UK Project).

We believe that such collaborations, and insights that stem from them, ultimately hold the key to the advancement of of energy-harvesting noise-sensing WSNs. Moreover, we suspect that the importance such an inter-disciplinary approach extends to other technical projects and endeavours that aim for significant societal relevance, not just energy-harvesting noise-sensing WSNs.

Bibliography

[1] Advanticsys. Advanticsys homepage, Nov. 2013. URL http://www.advanticsys.com/.

[2] J. Ai and A. A. Abouzeid. Coverage by directional sensors in randomly deployed wireless sensor networks. *Journal of Combinatorial Optimization*, 11(1):21–41, 2006. ISSN 1382-6905. doi: 10.1007/s10878-006-5975-x. URL http://dx.doi.org/10.1007/s10878-006-5975-x.

[3] C. Alippi, G. Anastasi, M. Di Francesco, and M. Roveri. Energy management in wireless sensor networks with energy-hungry sensors. *Instrumentation Measurement Magazine, IEEE*, 12(2):16–23, April 2009. ISSN 1094-6969. doi: 10.1109/MIM.2009.4811133.

[4] American National Standards Institute. *American national psychoacoustical terminology*. American National Standards Institute, New York, 1973.

[5] Analog Devices. ADP194: Logic controlled, high-side power switch, Sept. 2011. URL http://www.analog.com/.

[6] Analog Devices. ADMP401: Omnidirectional microphone with bottom port and analog output, June 2012. URL http://www.analog.com/media/en/technical-documentation/obsolete-data-sheets/ADMP401.pdf.

[7] Analog Devices. AN-1112: Microphone specifications explained, Mar. 2012. URL http://www.element14.com/community/servlet/JiveServlet/downloadBody/53446-102-2-269613/ADI.Application_Note_3.pdf.

[8] Analog Devices. MS-2348: Low self noise: The first step to high performance mems microphone ap-

plications, Aug. 2012. URL http://www.analog.com/media/en/technical-documentation/technical-articles/Low-Self-Noise-The-First-Step-to-High-Performance-MEMS-Microphone-Applications-MS-23pdf.

[9] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Netw.*, 7(3): 537–568, May 2009. ISSN 1570-8705. doi: 10.1016/j.adhoc.2008.06.003. URL http://dx.doi.org/10.1016/j.adhoc.2008.06.003.

[10] K. Astrom and T. Hagglund. *PID Controllers: Theory, Design and Tuning*, chapter 3. International Society of Automation (ISA), North Carolina, 2nd edition, 1995.

[11] S. Baghaee, H. Ulusan, S. Chamanian, O. Zorlu, H. Kulah, and E. Uysal-Biyikoglu. Towards a vibration energy harvesting wsn demonstration testbed. In *Digital Communications - Green ICT (TIWDC), 2013 24th Tyrrhenian International Workshop on*, pages 1–6, Sept 2013. doi: 10.1109/TIWDC.2013.6664202.

[12] M. Batty, K. W. Axhausen, F. Giannotti, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis, and Y. Portugali. Smart cities of the future. *The European Physical Journal Special Topics*, 214(1):481–518, 2012. ISSN 1951-6355. doi: 10.1140/epjst/e2012-01703-3. URL http://dx.doi.org/10.1140/epjst/e2012-01703-3.

[13] R. Beckwith, D. Teibel, and P. Bowen. Report from the field: results from an agricultural wireless sensor network. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 471–478, Nov 2004. doi: 10.1109/LCN.2004.105.

[14] D. A. Bies and C. H. Hansen. *Engineering Acoustics*. Spon Press, London, 2003.

[15] C. A. Boano, M. Zúñiga, J. Brown, U. Roedig, C. Keppitiyagama, and K. Römer. Templab: A testbed infrastructure to study the impact of temperature on wireless sensor networks. In *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, IPSN '14, pages 95–106, Piscataway, NJ, USA, 2014. IEEE Press. ISBN 978-1-4799-3146-0. URL `http://dl.acm.org/citation.cfm?id=2602339.2602351`.

[16] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 307–320, New York, NY, USA, 2006. ACM. ISBN 1-59593-343-3. doi: 10.1145/1182807.1182838. URL `http://doi.acm.org/10.1145/1182807.1182838`.

[17] Y. Cai, W. Lou, M. Li, and M. Li. Target-oriented scheduling in directional sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1550–1558, May 2007. doi: 10.1109/INFCOM.2007.182.

[18] D. Carli, D. Brunelli, D. Bertozzi, and L. Benini. A high-efficiency wind-flow energy harvester using micro turbine. In *Power Electronics Electrical Drives Automation and Motion (SPEEDAM), 2010 International Symposium on*, pages 778–783, June 2010. doi: 10.1109/SPEEDAM.2010.5542121.

[19] M. Chetto and H. Zhang. Performance evaluation of real-time scheduling heuristics for energy harvesting systems. In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, pages 398–403, New York, NY, USA, Dec 2010. ACM. doi: 10.1109/GreenCom-CPSCom.2010.16.

[20] R. Coughlin and F. Driscoll. *Operational Amplifiers and Linear Integrated Circuits*. Prentice Hall, London, 1987.

[21] J. C. Cuevas-Martinez, J. Canada-Bago, J. Fernandez-Prieto, and M. A. Gadeo-Martos. Knowledge-based duty cycle estimation in wireless sensor networks: Application for sound pressure monitoring. *Applied Soft Computing*, 13(2):967 – 980, 2013. ISSN 1568-4946. doi: http://dx.doi.org/10.1016/j.asoc.2012.10.005. URL `http://www.sciencedirect.com/science/article/pii/S156849461200453X`.

[22] Cymbet Corporation. Enerchip CC CBC3105, Aug. 2014. URL `http://www.cymbet.com/pdfs/DS-72-21.pdf`.

[23] Cymbet Corporation. CBC-EVAL-09: EnerChip EP universal energy harvester eval kit, Aug. 2014. URL `http://www.cymbet.com/pdfs/DS-72-13.pdf`.

[24] J. Dave, P. Halpern, and H. Myers. Computation of incident solar energy. *IBM Journal of Research and Development*, 19(6):539–549, Nov 1975. ISSN 0018-8646. doi: 10.1147/rd.196.0539.

[25] U. Department for Business Innovation and Skills. Smart cities: Background paper, October 2013. URL `https://www.gov.uk/government/publications/smart-cities-background-paper`.

[26] J. Djugash, S. Singh, G. Kantor, and W. Zhang. Range-only slam for robots operating cooperatively with sensor networks. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2078–2084, May 2006. doi: 10.1109/ROBOT.2006.1642011.

[27] M. Doddavenkatappa, M. Chan, and A. Ananda. Indriya: A low-cost, 3d wireless sensor network testbed. In T. Korakis, H. Li, P. Tran-Gia, and H.-S. Park, editors, *Testbeds and Research Infrastructure. Development of Networks and Communities*, volume 90 of *Lecture Notes of the*

Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 302–316. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29272-9. doi: 10.1007/978-3-642-29273-6_23. URL http://dx.doi.org/10.1007/978-3-642-29273-6_23.

[28] A. Dunkels. The ContikiMAC Radio Duty Cycling Protocol. In *SICS Technical Report*, 2011.

[29] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462, 2004. doi: 10.1109/LCN.2004.38.

[30] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis. Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 1–14, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0344-6. doi: 10.1145/1869983.1869985. URL http://doi.acm.org/10.1145/1869983.1869985.

[31] A. El-Hoiydi and J.-D. Decotignie. Wisemac: an ultra low power mac protocol for the downlink of infrastructure wireless sensor networks. In *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, volume 1, pages 244–251 Vol.1, 2004. doi: 10.1109/ISCC.2004.1358412.

[32] U. Energy Information Administration. Monthly Energy Review August 2015: Renewable Energy Production and Consumption by Source, Aug. 2015. URL http://www.eia.gov/totalenergy/data/monthly/pdf/sec10_3.pdf.

[33] European Commission. Directive 2002/49/EC of the European Parliament and of the Council of 25 June 2002 relating to the Assessment and

Management of Environmental Noise. In *Official Journal of the European Communities*, 2002.

[34] X. Fafoutis and N. Dragoni. Analytical comparison of mac schemes for energy harvesting - wireless sensor networks. In *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*, pages 1–6, 2012. doi: 10.1109/INSS.2012.6240580.

[35] L. Filipponi, S. Santini, and A. Vitaletti. Data collection in wireless sensor networks for noise pollution monitoring. In *Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems*, DCOSS '08, pages 492–497, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-69169-3. doi: 10.1007/978-3-540-69170-9_35. URL `http://dx.doi.org/10.1007/978-3-540-69170-9_35`.

[36] H. Fletcher and W. A. Munson. Loudness, its definition, measurement and calculation. *The Journal of the Acoustical Society of America*, 5(2):82–108, 1933. doi: 10.1121/1.1915637. URL `http://link.aip.org/link/?JAS/5/82/1`.

[37] R. Fonseca, P. Dutta, P. Levis, and I. Stoica. Quanto: Tracking energy in networked embedded systems. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, OSDI'08, pages 323–338, Berkeley, CA, USA, 2008. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1855741.1855764`.

[38] S. Franco. *Design with Operational Amplifiers and Analog Integrated Circuits*. McGraw-Hill, Singapore, 1988.

[39] G. Fusco and H. Gupta. Selection and orientation of directional sensors for coverage maximization. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on*, pages 1–9, June 2009. doi: 10.1109/SAHCN.2009.5168968.

[40] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W.H. Freeman and Company, New York, 1979.

[41] U. Government Office for Science. What are future cities? Origins, meanings and uses, June 2014. URL `https://www.gov.uk/government/publications/future-cities-origins-meanings-and-uses`.

[42] I. Hakala, M. Tikkakoski, and I. Kivelä. Wireless sensor network in environmental monitoring - case foxhouse. In *Proceedings of the 2008 Second International Conference on Sensor Technologies and Applications*, SENSORCOMM '08, pages 202–208, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3330-8. doi: 10.1109/SENSORCOMM.2008.27. URL `http://dx.doi.org/10.1109/SENSORCOMM.2008.27`.

[43] I. Hakala, I. Kivela, J. Ihalainen, J. Luomala, and C. Gao. Design of low-cost noise measurement sensor network: Sensor function design. In *Proceedings of the 2010 First International Conference on Sensor Device Technologies and Applications*, SENSORDEVICES '10, pages 172–179, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4094-8. doi: 10.1109/SENSORDEVICES.2010.39. URL `http://dx.doi.org/10.1109/SENSORDEVICES.2010.39`.

[44] I. Hakala, I. Kivela, J. Ihalainen, J. Luomala, and C. Gao. Design of noise measurement sensor network: Networking and communication part. In *Proceedings of the The Fifth International Conference on Sensor Technologies and Applications*, 2011.

[45] H. Hernandez, C. Blum, M. Middendorf, K. Ramsch, and A. Scheidler. Self-synchronized duty-cycling for mobile sensor networks with energy harvesting capabilities: A swarm intelligence study. In *Swarm Intelligence Symposium, 2009. SIS '09. IEEE*, pages 153–159, 2009. doi: 10.1109/SIS.2009.4937858.

[46] H. Hernandez, M. J. Blesa, C. Blum, T. Baumgartner, S. P. Fekete, and

A. Kroller. A protocol for self-synchronized duty-cycling in sensor networks: Generic implementation in wiselib. In *Proceedings of the 2010 Sixth International Conference on Mobile Ad-hoc and Sensor Networks*, MSN '10, pages 134–139, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4315-4. doi: 10.1109/MSN.2010.27. URL http://dx.doi.org/10.1109/MSN.2010.27.

[47] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11): 93–104, Nov. 2000. ISSN 0362-1340. doi: 10.1145/356989.356998. URL http://doi.acm.org/10.1145/356989.356998.

[48] J. L. Hill and D. E. Culler. Mica: a wireless platform for deeply embedded networks. *Micro, IEEE*, 22(6):12–24, Nov 2002. ISSN 0272-1732. doi: 10.1109/MM.2002.1134340.

[49] J. Hsu, S. Zahedi, A. Kansal, M. Srivastava, and V. Raghunathan. Adaptive duty cycling for energy harvesting systems. In *Proceedings of the 2006 International Symposium on Low Power Electronics and Design*, ISLPED '06, pages 180–185, New York, NY, USA, 2006. ACM. ISBN 1-59593-462-6. doi: 10.1145/1165573.1165616. URL http://doi.acm.org/10.1145/1165573.1165616.

[50] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange. Sensorscope: Application-specific sensor network for environmental monitoring. *ACM Trans. Sen. Netw.*, 6(2):17:1–17:32, Mar. 2010. ISSN 1550-4859. doi: 10.1145/1689239.1689247. URL http://doi.acm.org/10.1145/1689239.1689247.

[51] T. Instruments. MSP430x1xx family user's guide, Apr. 2015. URL http://www.ti.com/lit/ug/slau049f/slau049f.pdf.

[52] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans.*

Netw., 11(1):2–16, Feb. 2003. ISSN 1063-6692. doi: 10.1109/TNET.2002.808417. URL http://dx.doi.org/10.1109/TNET.2002.808417.

[53] Internet Engineering Task Force: Mobile Ad hoc Networks Working Group. Dynamic MANET On-demand (AODVv2) Routing (draft-ietf-manet-dymo-26), Aug. 2013. URL http://tools.ietf.org/html/draft-ietf-manet-dymo-26.

[54] V. Iyer, S. S. Iyengar, N. Balakrishnan, V. Phoha, and M. B. Srinivas. Farms: Fusionable ambient renewable macs. In *Sensors Applications Symposium, 2009. SAS 2009. IEEE*, pages 169–174, 2009. doi: 10.1109/SAS.2009.4801800.

[55] J. Jeong, X. Jiang, and D. Culler. Design and analysis of micro-solar power systems for wireless sensor networks. In *Networked Sensing Systems, 2008. INSS 2008. 5th International Conference on*, pages 181–188, New York, NY, USA, June 2008. IEEE. doi: 10.1109/INSS.2008.4610922.

[56] X. Jiang, P. Dutta, D. Culler, and I. Stoica. Micro power meter for energy monitoring of wireless sensor networks at scale. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 186–195, New York, NY, USA, April 2007. IEEE. doi: 10.1109/IPSN.2007.4379678.

[57] A. Kansal, D. Potter, and M. B. Srivastava. Performance aware tasking for environmentally powered sensor networks. *SIGMETRICS Perform. Eval. Rev.*, 32(1):223–234, June 2004. ISSN 0163-5999. doi: 10.1145/1012888.1005714. URL http://doi.acm.org/10.1145/1012888.1005714.

[58] A. Kansal, J. Hsu, M. B. Srivastava, and V. Raqhunathan. Harvesting aware power management for sensor networks. In *Design Automation Conference, 2006 43rd ACM/IEEE*, pages 651–656, 2006. doi: 10.1109/DAC.2006.229276.

[59] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power management in energy harvesting sensor networks. *ACM Trans. Embed. Comput. Syst.*, 6(4), Sept. 2007. ISSN 1539-9087. doi: 10.1145/1274858.1274870. URL `http://doi.acm.org/10.1145/1274858.1274870`.

[60] J. A. Khan, H. K. Qureshi, and A. Iqbal. Energy management in wireless sensor networks: A survey. *Computers & Electrical Engineering*, 41:159 – 176, 2015. ISSN 0045-7906. doi: http://dx.doi.org/10.1016/j.compeleceng.2014.06.009. URL `http://www.sciencedirect.com/science/article/pii/S0045790614001773`.

[61] H. Kim, H. Lee, and S. Lee. A cross-layer optimization for energy-efficient mac protocol with delay and rate constraints. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 2336–2339, 2011. doi: 10.1109/ICASSP.2011.5946951.

[62] J. Kim and J.-W. Lee. Performance analysis of the energy adaptive mac protocol for wireless sensor networks with rf energy transfer. In *ICT Convergence (ICTC), 2011 International Conference on*, pages 14–19, 2011. doi: 10.1109/ICTC.2011.6082542.

[63] P. W. Lee, M. Han, H.-P. Tan, and A. Valera. An empirical study of harvesting-aware duty cycling in environmentally-powered wireless sensor networks. In *Communication Systems (ICCS), 2010 IEEE International Conference on*, pages 306–310, New York, NY, USA, Nov 2010. IEEE. doi: 10.1109/ICCS.2010.5686442.

[64] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 126–137, New York, NY, USA, 2003. ACM. ISBN 1-58113-707-9. doi: 10.1145/958491.958506. URL `http://doi.acm.org/10.1145/958491.958506`.

[65] S. Liu, Q. Qiu, and Q. Wu. Energy aware dynamic voltage and frequency selection for real-time systems with energy harvesting. In *Design, Automation and Test in Europe, 2008. DATE '08*, pages 236–241, New York, NY, USA, March 2008. IEEE. doi: 10.1109/DATE.2008.4484692.

[66] H. Ma and Y. Liu. On coverage problems of directional sensor networks. In X. Jia, J. Wu, and Y. He, editors, *Mobile Ad-hoc and Sensor Networks*, volume 3794 of *Lecture Notes in Computer Science*, pages 721–731. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30856-0. doi: 10.1007/11599463_70. URL `http://dx.doi.org/10.1007/11599463_70`.

[67] H. Ma and Y. Liu. Some problems of directional sensor networks. *Int. J. Sen. Netw.*, 2(1/2):44–52, Apr. 2007. ISSN 1748-1279. doi: 10.1504/IJSNET.2007.012981. URL `http://dx.doi.org/10.1504/IJSNET.2007.012981`.

[68] M. Malinowski, M. Moskwa, M. Feldmeier, M. Laibowitz, and J. A. Paradiso. Cargonet: A low-cost micropower sensor node exploiting quasi-passive wakeup for adaptive asychronous monitoring of exceptional events. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, SenSys '07, pages 145–159, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-763-6. doi: 10.1145/1322263.1322278. URL `http://0-doi.acm.org/10.1145/1322263.1322278`.

[69] J. C. A. Mariscal-Ramirez, J. A. Fernandez-Prieto, M. A. Gadeo-Martos, and J. Canada-Bago. Knowledge-based wireless sensors using sound pressure level for noise pollution monitoring. In *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pages 1032–1037, 2011. doi: 10.1109/ISDA.2011.6121794.

[70] T. Markvart, editor. *Solar Electricity*, chapter 2. John Wiley and Sons, West Sussex, England, 2nd edition, 2000.

[71] T. Markvart, editor. *Solar Electricity*, chapter 3. John Wiley and Sons, West Sussex, England, 2nd edition, 2000.

[72] Arduino. Arduino, Nov. 2013. URL http://www.arduino.cc/.

[73] BBC. Copenhagen: Noise pollution ruling in denmark, March 2014. URL http://www.bbc.co.uk/news/world-europe-26404666.

[74] Casella. An introduction to industrial noise measurements and hearing issues in the workplace, Feb. 2010. URL http://www.enviroequipment.com/rentals/documents/cel_noise_book.pdf.

[75] Casella USA. Cel-240 digital sound level meter, November 2013. URL http://www.enviroequipment.com/rentals/documents/CEL-240UserManual.pdf.

[76] Center for Urban Science + Progress. NYU Center for Urban Science and Progress, Jan. 2015. URL http://cusp.nyu.edu/.

[77] Centre for Advanced Spatial Analysis (CASA), University College London. Citydashboard: London, March 2014. URL http://citydashboard.org/london/.

[78] Cooper Bussmann. Cooper Bussmann PowerStor supercapacitors: HV Series, Jan. 2014. URL http://www.cooperindustries.com/content/dam/public/bussmann/Electronics/Resources/product-datasheets/Bus_Elx_DS_4376_HV_Series.pdf.

[79] Cymbet Corporation. AN-1008: Using the EnerChip batteries instead of coin cells and super capacitors, Apr. 2012. URL http://www.cymbet.com/pdfs/AN-1008-Replacing-Coin-Cells-and-Supercapacitors.pdf.

[80] Cymbet Corporation. AN-1025: Using the EnerChip in pulse current applications, Apr. 2012. URL http://www.cymbet.com/pdfs/AN-1025.pdf.

[81] Cymbet Corporation. AN-1036: Using the EnerChip CC in energy harvesting designs, Apr. 2012. URL http://www.cymbet.com/pdfs/AN-1036.pdf.

[82] Cymbet Corporation. 6 reasons why EnerChips uniquely enable green solutions, Apr. 2012. URL http://www.cymbet.com/pdfs/Cymbet-Eco-Friendly-Products-6-reasons-RevA.pdf.

[83] Cymbet Corporation. CBC915: EnerChip EP energy processor, Aug. 2014. URL http://www.cymbet.com/pdfs/DS-72-15.pdf.

[84] Hamamatsu. S1087/S1133 series: Ceramic package photodiode with low dark current, Apr. 2011. URL http://www.advanticsys.com/shop/documents/1320249860_Light_Hamamatsu_S1087.pdf.

[85] Innovate UK. Innovate UK - GOV.UK, Jan. 2012. URL https://www.gov.uk/government/organisations/innovate-uk.

[86] Keithley Instruments. 2401 Low Voltage Sourcemeter Instrument, Jan. 2012. URL http://www.keithley.com/data?asset=55849.

[87] Libelium. Smart Cities Technical Guide, Nov. 2012. URL http://www.libelium.com/v11-files/documentation/waspmote/smart-cities-sensor-board_eng.pdf.

[88] Libelium. Waspmote Datasheet, June 2015. URL http://www.libelium.com/development/waspmote/documentation/waspmote-datasheet/?action=download.

[89] Linear Technology. LT1615/LT1615-1: Micropower step-up dc/dc converters in thinsot, Mar. 1998. URL http://cds.linear.com/docs/en/datasheet/16151fas.pdf.

[90] Linear Technology. LT6105: Precision, extended input range current sense amplifier, Nov. 2013. URL http://cds.linear.com/docs/en/datasheet/6105fa.pdf.

[91] Lumileds. LUXEON K: Plug and play matrix solution with precise flux, V_f and color, June 2014. URL http://www.lumileds.com/uploads/366/DS102-pdf.

[92] Maxim. MAX323/MAX324/MAX325: Precision, single-supply, spst analog switches, Oct. 1997. URL http://datasheets.maximintegrated.com/en/ds/MAX323-MAX325.pdf.

[93] National Instruments. LabVIEW System Design Software, Dec. 2014. URL http://www.ni.com/labview/.

[94] Nest Labs. Nest, April 2015. URL http://nest.com/.

[95] New York City Government. New York City Noise Code (Local law 113 of 2005), Jan. 2005. URL http://www.nyc.gov/html/dep/html/noise/index.shtml.

[96] Oracle Labs. Sun spot world, Nov. 2013. URL http://www.sunspotdev.org/.

[97] Panasonic. Nickel Cadmium Batteries Technical Handbook, Jan. 2000. URL http://www.battery-kutter.de/out/basic/src/pdf/panasonic/nicd.pdf.

[98] Panasonic. WM-55A103/WM-56A103: Unidirectional Back Electret Condenser Microphone Cartridge, July 2004. URL https://courses.cit.cornell.edu/ee476/FinalProjects/s2007/ai45_hkc2_sbs43/ai45_hkc2_sbs43/em10_wm55(6)a103_dne.pdf.

[99] Pico Technology. Picolog 1000 series: User's guide, May 2013. URL https://www.picotech.com/download/manuals/pl1000-en-2.pdf.

[100] PolySolar Limited. Polysolar, Jan. 2014. URL http://www.polysolar.co.uk/.

[101] PowerStream. PowerStream Li-ion Button Cell Lir2450 Data Sheet, Mar. 2008. URL http://www.powerstream.com/p/Lir2450.pdf.

[102] SparkFun Electronics. Sparkfun homepage, November 2013. URL https://www.sparkfun.com/.

[103] TinyOS. TinyOS Documentation Wiki, May 2013. URL http://tinyos.stanford.edu/tinyos-wiki/index.php/Main_Page.

[104] TinyOS Core Working Group. TinyOS TEP 105: Low Power Listening, Nov. 2007. URL http://www.tinyos.net/tinyos-2.x/doc/html/tep105.html.

[105] TinyOS Core Working Group. Tymo, Apr. 2008. URL http://tinyos.stanford.edu/tinyos-wiki/index.php/Tymo.

[106] Touchstone Semiconductor. TS1100: A 1μA, +2V to +25V SOT23 precision current-sense amplifier, Nov. 2013. URL http://www.silabs.com/SupportDocuments/TechnicalDocs/TS1101.pdf.

[107] A. B. Meinel and M. P. Meinel. *Applied Solar Energy: An Introduction*. Addison-Wesley, Massachusetts, USA, 1st edition, 1976.

[108] R. A. Messenger and J. Ventre. *Photovoltaic Systems Engineering*, chapter 2. CRC Press, Florida, USA, 2nd edition, 2010.

[109] R. A. Messenger and J. Ventre. *Photovoltaic Systems Engineering*, chapter 10. CRC Press, Florida, USA, 2nd edition, 2010.

[110] R. A. Messenger and J. Ventre. *Photovoltaic Systems Engineering*, chapter 10. CRC Press, Florida, USA, 2nd edition, 2010.

[111] R. Morais, S. G. Matos, M. A. Fernandes, A. L. Valente, S. F. Soares, P. Ferreira, and M. Reis. Sun, wind and water flow as energy supply for small stationary data acquisition platforms. *Computers and Electronics in Agriculture*, 64(2):120 – 132, 2008. ISSN 0168-1699. doi: http://dx.doi.org/10.1016/j.compag.2008.04.005. URL http://www.sciencedirect.com/science/article/pii/S0168169908001257.

[112] C. Moser, D. Brunelli, L. Thiele, and L. Benini. Real-time scheduling with regenerative energy. In *Real-Time Systems, 2006. 18th Euromicro Conference on*, pages 260–270, New York, NY, USA, 2006. IEEE. doi: 10.1109/ECRTS.2006.23.

[113] M. Moser. *Engineering Acoustics*. Springer, London, 2nd edition, 2009.

[114] V. P. Munishwar and N. B. Abu-Ghazaleh. Coverage algorithms for visual sensor networks. *ACM Trans. Sen. Netw.*, 9(4):45:1–45:36, July 2013. ISSN 1550-4859. doi: 10.1145/2489253.2489262. URL `http://doi.acm.org/10.1145/2489253.2489262`.

[115] D. Musiani, K. Lin, and T. S. Rosing. Active sensing platform for wireless structural health monitoring. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 390–399, April 2007. doi: 10.1109/IPSN.2007.4379699.

[116] W. H. Organization. Global health observatory (GHO): Urban population growth, January 2014. URL `http://www.who.int/gho/urban_health/situation_trends/urban_population_growth_text/en/`.

[117] Oriel Instruments. Oriel Sol3a Class AAA Solar Simulators, July 2012. URL `http://assets.newport.com/webDocuments-EN/images/DS-12082_Sol3_Solar_Sim.pdf`.

[118] C. Park and P. Chou. Power utility maximization for multiple-supply systems by a load-matching switch. In *Low Power Electronics and Design, 2004. ISLPED '04. Proceedings of the 2004 International Symposium on*, pages 168–173, Aug 2004. doi: 10.1109/LPE.2004.1349329.

[119] S. Park, A. Savvides, and M. B. Srivastava. Sensorsim: A simulation framework for sensor networks. In *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '00, pages 104–111, New York, NY, USA,

2000. ACM. ISBN 1-58113-304-9. doi: 10.1145/346855.346870. URL http://0-doi.acm.org/10.1145/346855.346870.

[120] S. Park, A. Savvides, and M. B. Srivastava. Simulating networks of wireless sensors. In *Simulation Conference, 2001. Proceedings of the Winter*, volume 2, pages 1330–1338 vol.2, New York, NY, USA, 2001. IEEE. doi: 10.1109/WSC.2001.977453.

[121] K. S. J. Pister and L. Doherty. Tsmp: Time synchronized mesh protocol. In *In Proceedings of the IASTED International Symposium on Distributed Sensor Networks (DSN08*, 2008.

[122] P. Poggi, G. Notton, M. Muselli, and A. Louche. Stochastic study of hourly total solar radiation in corsica using a markov model. *International Journal of Climatology*, 20(14):1843–1860, 2000. ISSN 1097-0088. doi: 10.1002/1097-0088(20001130)20:14<1843::AID-JOC561>3.0.CO;2-O. URL http://dx.doi.org/10.1002/1097-0088(20001130)20:14<1843::AID-JOC561>3.0.CO;2-O.

[123] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, IPSN '05, Piscataway, NJ, USA, 2005. IEEE Press. ISBN 0-7803-9202-7. URL http://dl.acm.org/citation.cfm?id=1147685.1147744.

[124] J. Porter, P. Arzberger, H.-W. Braun, P. Bryant, S. Gage, T. Hansen, P. Hanson, C.-C. Lin, F.-P. Lin, T. Kratz, W. Michener, S. Shapiro, and T. Williams. Wireless sensor networks for ecology. *BioScience*, 55(7): 561–572, 2005. doi: 10.1641/0006-3568(2005)055[0561:WSNFE]2.0.CO;2. URL http://bioscience.oxfordjournals.org/content/55/7/561.abstract.

[125] V. Raghunathan, P. Spanos, and M. B. Srivastava. Adaptive power-fidelity in energy-aware wireless embedded systems. In *Real-Time Systems Sym-*

posium, 2001. (RTSS 2001). Proceedings. 22nd IEEE, pages 106–115, New York, NY, USA, Dec 2001. ACM. doi: 10.1109/REAL.2001.990601.

[126] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava. Cyclops: In situ image sensing and interpretation in wireless sensor networks. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, SenSys '05, pages 192–204, New York, NY, USA, 2005. ACM. ISBN 1-59593-054-X. doi: 10.1145/1098918.1098939. URL `http://doi.acm.org/10.1145/1098918.1098939`.

[127] L. Rizzon, M. Rossi, R. Passerone, and D. Brunelli. Wireless sensor networks for environmental monitoring powered by microprocessors heat dissipation. In *Proceedings of the 1st International Workshop on Energy Neutral Sensing Systems*, ENSSys '13, pages 8:1–8:6, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2432-8. doi: 10.1145/2534208.2534216. URL `http://doi.acm.org/10.1145/2534208.2534216`.

[128] C. Rusu, R. Melhem, and D. Mossé. Multi-version scheduling in rechargeable energy-aware real-time systems. *J. Embedded Comput.*, 1(2):271–283, Apr. 2005. ISSN 1740-4460. URL `http://dl.acm.org/citation.cfm?id=1233760.1233769`.

[129] S. Santini and A. Vitaletti. Wireless sensor networks for environmental noise monitoring, Jul 2007. URL `http://www.vs.inf.ethz.ch/publ/papers/santinis07_noise.pdf`.

[130] S. Santini, B. Ostermaier, and A. Vitaletti. First experiences using wireless sensor networks for noise pollution monitoring. In *Proceedings of the workshop on Real-world wireless sensor networks*, REALWSN '08, pages 61–65, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-123-1. doi: 10.1145/1435473.1435490. URL `http://doi.acm.org/10.1145/1435473.1435490`.

[131] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu,

W. Kang, J. Stankovic, D. Young, and J. Porter. Luster: Wireless sensor network for environmental research. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, SenSys '07, pages 103–116, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-763-6. doi: 10.1145/1322263.1322274. URL http://doi.acm.org/10.1145/1322263.1322274.

[132] Sentec. Homepage - Sentec, Jan. 2012. URL http://www.sentec.co.uk/.

[133] Sentec. Keeping the noise down: autonomous sensors, Aug. 2015. URL http://www.sentec.co.uk/newsandthinking/news/solar-owl.

[134] V. Sharma, U. Mukherji, and V. Joseph. Efficient energy management policies for networks with energy harvesting sensor nodes. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 375–383, 2008. doi: 10.1109/ALLERTON.2008.4797582.

[135] Y. Shi and Y. T. Hou. Optimal base station placement in wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(4):32:1–32:24, Nov. 2009. ISSN 1550-4859. doi: 10.1145/1614379.1614384. URL http://doi.acm.org/10.1145/1614379.1614384.

[136] V. Shnayder, M. Hempstead, B.-r. Chen, G. W. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 188–200, New York, NY, USA, 2004. ACM. ISBN 1-58113-879-2. doi: 10.1145/1031495.1031518. URL http://0-doi.acm.org/10.1145/1031495.1031518.

[137] A. Sinha and A. Chandrakasan. Dynamic power management in wireless sensor networks. *IEEE Des. Test*, 18(2):62–74, Mar. 2001. ISSN 0740-7475. doi: 10.1109/54.914626. URL http://dx.doi.org/10.1109/54.914626.

[138] W. D. Stanley. *Operational Amplifiers with Linear Integrated Circuits*. Merrill, New York, 1994.

[139] I. Stark. Converting body heat into reliable energy for powering physiological wireless sensors. In *Proceedings of the 2nd Conference on Wireless Health*, WH '11, pages 31:1–31:2, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0982-0. doi: 10.1145/2077546.2077580. URL http://0-doi.acm.org/10.1145/2077546.2077580.

[140] B. G. Streetman. *Solid State Electronic Devices*. Prentice-Hall, New Jersey, USA, 4th edition, 1995.

[141] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 214–226, New York, NY, USA, 2004. ACM. ISBN 1-58113-879-2. doi: 10.1145/1031495.1031521. URL http://doi.acm.org/10.1145/1031495.1031521.

[142] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Commun. ACM*, 47 (6):34–40, June 2004. ISSN 0001-0782. doi: 10.1145/990680.990704. URL http://0-doi.acm.org/10.1145/990680.990704.

[143] W. Tan and S. Jarvis. On the design of an energy-harvesting noise-sensing wsn mote. *EURASIP Journal on Wireless Communications and Networking*, 2014(1):167, 2014. ISSN 1687-1499. doi: 10.1186/1687-1499-2014-167. URL http://jwcn.eurasipjournals.com/content/2014/1/167.

[144] W. M. Tan and S. A. Jarvis. Self-determination of maximum supportable receiver wakeup intervals in energy harvesting wsn nodes. In *Wireless Days (WD), 2013 IFIP*, pages 1–7, Nov 2013. doi: 10.1109/WD.2013.6686440.

[145] W. M. Tan and S. A. Jarvis. Energy harvesting noise pollution sensing wsn mote: Survey of capabilities and limitations. In *Wireless Sensor (ICWISE), 2013 IEEE Conference on*, pages 53–60, Dec 2013. doi: 10.1109/ICWISE.2013.6728779.

[146] W. M. Tan and S. A. Jarvis. Determination of maximum supportable receiver wakeup intervals in energy harvesting wsn nodes using a client-server setup. In *Wireless Sensor (ICWISE), 2013 IEEE Conference on*, pages 61–67, Dec 2013. doi: 10.1109/ICWISE.2013.6728780.

[147] W. M. Tan and S. A. Jarvis. Quality over quantity: Target identifiability in directional sensor networks. In *Wireless Sensors (ICWiSE), 2014 IEEE Conference on*, pages 41–48, Oct 2014. doi: 10.1109/ICWISE.2014.7042659.

[148] W. M. Tan and S. A. Jarvis. A distributed heuristic solution to the target identifiability problem in directional sensor networks. In *Computing, Networking and Communications (ICNC), 2015 International Conference on*, pages 176–182, Feb 2015. doi: 10.1109/ICCNC.2015.7069337.

[149] J. Taneja, J. Jeong, and D. Culler. Design, modeling, and capacity planning for micro-solar power sensor networks. In *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, pages 407–418, New York, NY, USA, April 2008. ACM. doi: 10.1109/IPSN.2008.67.

[150] D. Tao, S. Tang, and L. Liu. Constrained artificial fish-swarm based area coverage optimization algorithm for directional sensor networks. In *Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE 10th International Conference on*, pages 304–309, 2013. doi: 10.1109/MASS.2013.89.

[151] Texas Instruments. MSP430x12x mixed signal microcontroller datasheet, Sept. 2004. URL `http://www.ti.com/lit/ds/slas312c/slas312c.pdf`.

[152] Texas Instruments. Opa344/opa345: Low power, single-supply, rail-to-rail operational amplifiers: Microamplifier series, Aug. 2008. URL http://www.ti.com/lit/ds/symlink/opa344.pdf.

[153] Texas Instruments. eZ430-RF2500-SEH: Solar energy harvesting development kit, Mar. 2009. URL http://www.ti.com/lit/ml/sprt506/sprt506.pdf.

[154] Texas Instruments. LM3406: 1.5-A, Constant current, buck regulator for driving high power LEDs, Nov. 2013. URL http://www.ti.com/lit/ds/symlink/lm3406.pdf.

[155] Texas Instruments. CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF transceiver datasheet, Oct. 2014. URL http://www.ti.com/lit/ds/symlink/cc2420.pdf.

[156] M. Varshney, D. Xu, M. Srivastava, and R. Bagrodia. squalnet: A scalable simulation and emulation environment for sensor networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, IPSN/SPOTS '07, New York, NY, USA, 2007. ACM.

[157] G. Vellidis, M. Tucker, C. Perry, C. Kvien, and C. Bednarz. A real-time wireless smart sensor array for scheduling irrigation. *Computers and Electronics in Agriculture*, 61(1):44 – 50, 2008. ISSN 0168-1699. doi: http://dx.doi.org/10.1016/j.compag.2007.05.009. URL http://www.sciencedirect.com/science/article/pii/S0168169907001706. Emerging Technologies For Real-time and Integrated Agriculture Decisions.

[158] H. Visser and R. Vullers. Rf energy harvesting and transport for wireless sensor network applications: Principles and requirements. *Proceedings of the IEEE*, 101(6):1410–1423, June 2013. ISSN 0018-9219. doi: 10.1109/JPROC.2013.2250891.

[159] J. Wang, C. Niu, and R. Shen. Priority-based target coverage in directional sensor networks using a genetic algorithm. *Computers & Math-*

ematics with Applications, 57(11):1915 – 1922, 2009. ISSN 0898-1221. doi: http://dx.doi.org/10.1016/j.camwa.2008.10.019. URL http://www.sciencedirect.com/science/article/pii/S0898122108005336. Proceedings of the International Conference Bio-Inspired Computing-Theories and Applications BIC-TA 2007 Zhengzhou, China.

[160] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: A wireless sensor network testbed. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, IPSN '05, Piscataway, NJ, USA, 2005. IEEE Press. ISBN 0-7803-9202-7. URL http://dl.acm.org/citation.cfm?id=1147685.1147769.

[161] H. Yang, D. Li, and H. Chen. Coverage quality based target-oriented scheduling in directional sensor networks. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5, May 2010. doi: 10.1109/ICC.2010.5501996.

[162] J. Yang, O. Ozel, and S. Ulukus. Optimal packet scheduling in a broadcast channel with an energy harvesting transmitter. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5, 2011. doi: 10.1109/icc.2011.5963217.

[163] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi. Hardware design experiences in zebranet. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 227–238, New York, NY, USA, 2004. ACM. ISBN 1-58113-879-2. doi: 10.1145/1031495.1031522. URL http://doi.acm.org/10.1145/1031495.1031522.

[164] W. Zhang, G. Kantor, and S. Singh. Integrated wireless sensor/actuator networks in an agricultural application. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 317–317, New York, NY, USA, 2004. ACM. ISBN 1-58113-879-

2. doi: 10.1145/1031495.1031560. URL http://doi.acm.org/10.1145/1031495.1031560.